ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Estudio del estándar de televisión digital interactiva HbbTV e implementación de aplicación final

Proyecto final de carrera

Ingeniería técnica de telecomunicación, especialidad en sonido e imagen

Imanol Eslava Arce

Tutor: Mikel Sagüés García

Pamplona, 28 de febrero de 2014

ÍNDICE

INTRODUCCIÓN

La televisión conectada se está erigiendo como uno de los fenómenos tecnológicos más destacados de los últimos años en el campo de las telecomunicaciones. A nivel europeo cabe destacar la importancia del estándar Hybrid Broadcast Broadband TV, también conocido como HbbTV. Se trata de una plataforma abierta que tiene el objetivo de proporcionar acceso a aplicaciones interactivas y a contenidos bajo demanda de los consumidores, independientemente de la marca del terminal o dispositivo de que dispongan.

Las perspectivas de éxito de implantación de este estándar son relativamente altas respecto a anteriores intentos, debido principalmente a que no se trata de una nueva tecnología que conlleve unos costes elevados de desarrollo, sino que HbbTV es una conjunción de la actual tecnología de difusión de la televisión digital con Internet. De hecho, en Europa, tres de cada diez televisores que se venden incluye acceso HbbTV con un fuerte tirón en Alemania (con una penetración en venta del 53%), España (26%) y Francia (21%).

Como se puede ver, la televisión híbrida se encuentra en expansión y de hecho ya se está tratando de expandir por el resto del mundo y no solo en Europa, en lugares como Australia, Malasia y Vietnam.

En este PFC, se ha trabajado con esta tecnología desde su punto inicial (antecedentes, estado del arte, especificaciones…) hasta su punto final (creación de una aplicación HbbTV).

El proyecto ha quedado dividido en cinco grandes capítulos:

El primero de ellos se encarga de explicar el porqué de esta tecnología (necesidad de interacción con la televisión), los antecedentes a esta tecnología y sus competidores o complementarios actuales. Además, en este capítulo se define que es HbbTV, su situación actual en el mundo y en España, y el estado del mercado actual y las opciones que se disponen en la actualidad para poder trabajar con esta tecnología.

El segundo de los capítulos trata de explicar los servicios de radiodifusión (broadband) y de banda ancha (broadband) ya que tal y como se ha dicho, HbbTV es una plataforma de emisión de contenidos bajo demanda combinando estos dos servicios. En este capítulo se define el estándar HbbTV y todas sus especificaciones. Su funcionamiento, sus tipos de aplicaciones, escenarios, tecnologías utilizadas…

La tercera parte de este proyecto consiste en la toma de contacto con la televisión híbrida. Estudiar los lenguajes de programación necesarios para poder crear aplicaciones, visualizar aplicaciones existentes, ver el código de

esas aplicaciones, realizar modificaciones en dichos códigos y crear pequeñas aplicaciones propias han sido las acciones principales realizadas en esta parte.

El cuarto capítulo es una práctica para una asignatura llamada Tecnologías e Instalaciones de Vídeo que pertenece al Grado en Ingeniería en Tecnologías de Telecomunicación, en la cual uno de los temas era este de televisión híbrida. Surgió la oportunidad de crear una práctica para este tema, de forma que los alumnos pudieran conocer el código de una aplicación y pudieran realizar pequeñas modificaciones sobre dicho código para realizar la práctica. Se creó una aplicación HbbTV, se documentó el código y se dio a los alumnos las pautas necesarias para iniciarse en el mundo de la televisión híbrida.

Para terminar, en la última de las partes del proyecto se creó una aplicación HbbTV. Se buscaba que pudiera ser un caso real, una aplicación que un usuario se pudiera encontrar en un mercado de aplicaciones, por lo que el punto de partida fue un usuario en frente de su televisor deseando ver una película en concreto. El trabajo consiste en una aplicación a modo de vídeo club, de forma que el usuario pueda elegir el tipo de género y dentro del tipo de género la película que quiere visualizar. Para poder ver la película, dicho usuario tiene que estar registrado en la base de datos del vídeo club, por lo que si no introduce unos datos correctos en la llamada a base de datos no puede ver la película. Para su desarrollo se ha hecho uso de los emuladores que algunos navegadores como Opera o Firefox ofrecen. En el caso de Firefox, este navegador ofrece un complemento que permite visualizar las aplicaciones HbbTV con tan solo copiar su url. En el caso de Opera, a parte del paquete que ofrece dicho navegador es necesaria la descarga de una caja virtual para poder volcar ahí las aplicaciones HbbTV.

.

CAPÍTULO 1-HBBTV

Cuando el ordenador se come paulatinamente el mercado de entretenimiento audiovisual, las ventas de televisores se ven seriamente afectadas. Todo el mundo relacionado con la venta de televisores ve necesaria una medida para poder seguir viviendo de su trabajo. La solución tomada es la de la convergencia.

Dicha convergencia consistiría en aunar la experiencia multimedia, introduciendo tanto contenido en broadcast como hasta la fecha y añadiéndole contenido en internet de forma que se pudiera producir una interactividad con el televisor demandado por el propio usuario.

Esta interacción se realizaría con el propio mando de televisión, y se trataría de realizar mediante una interfaz unificada.

El objetivo sería convertir la televisión en un ordenador de propósito específico.

1.1. Interactividad

La interactividad es la capacidad de ofrecer contenidos adicionales a los programas de televisión, permitiendo al usuario ver informaciones asociadas al contenido audiovisual, la programación de los canales, participar en concursos, votaciones, comprar productos o servicios, e incluso participar en los propios programas de televisión con el mando a distancia. La interactividad es posible gracias a aplicaciones que complementan la programación, siendo el usuario el que decide si quiere o no verlos, y cuándo verlos.

La interactividad ofrece al espectador la posibilidad de personalizar el contenido que muestra su televisor, bien sea accediendo a información enviada durante el proceso de emisión pero que sólo se hace visible si el espectador lo desea, o bien accediendo a servidores con los que puede intercambiar información, a través de un canal de retorno utilizando el televisor como interfaz de salida.

La interactividad va a permitir a los canales de televisión ofrecer un importante conjunto de servicios al ciudadano, que permitan explorar nuevas formas de hacer televisión, incorporando funciones avanzadas de comunicación y participación, y servicios sociales para el desarrollo de la Sociedad de la Información. Por el lado de los usuarios, la interactividad va a permitir acceder a nuevos contenidos, a una televisión mucho más rica y completa, con la posibilidad de participar e influir en los programas de televisión.

La interactividad permite complementar los contenidos de televisión, tanto a través de servicios públicos (ayuntamientos, gobiernos, sanidad, sectores desprotegidos, etc.) como servicios comerciales o de entretenimiento

(votaciones, concursos, publicidad interactiva, etc.) que hasta ahora solo eran accesibles a través de otros medios como ordenador o teléfono móvil.

La principal ventaja de la interactividad en televisión, radica en la posibilidad de acceder a un amplio conjunto de servicios públicos o privados a través del televisor, con un único terminal y un mando a distancia. Otra ventaja de la interactividad radica en que es el propio usuario el que decide si quiere o no ver los servicios interactivos y los contenidos asociados a la interactividad (por ejemplo, si quiere o no ver los mensajes que los usuarios envían a los programas tipo SMS). Finalmente, la interactividad en televisión permite ofrecer servicios adaptados a las necesidades de los diferentes colectivos que conforman la sociedad, independientemente de la edad y la localización.

1.2. Antecedentes

Antes de la creación de HbbTV, ya aparecieron otras formas de interactividad en televisión.

1.2.1. MHP

MHP (Multimedia Home Platform) [MHP] es un sistema intermedio (middleware en inglés), diseñado por el proyecto DVB [DVB] y estandarizado por la ETSI. MHP define una plataforma común para las aplicaciones interactivas de la televisión digital, independiente tanto del proveedor de servicios interactivos como del receptor de televisión utilizado. De este modo, MHP favorece la creación de un mercado horizontal donde aplicaciones, red de transmisión y terminales MHP pueden ser suministrados por proveedores o fabricantes independientes.

El estándar MHP soporta distintos tipos de aplicaciones interactivas:

-Guía Electrónica de Programas (EPG).

-Servicios de información como noticias, deportes, teletexto…

-Aplicaciones sincronizadas con el contenido de los programas.

-E-mail y acceso a Internet.

-Otros servicios: comercio electrónico, servicios públicos de educación y salud…

MHP define un interfaz genérico entre las aplicaciones digitales interactivas proporcionadas por DVB y los terminales en los cuales se van a ejecutar (set top box, IRD's, PC's multimedia…).

El objetivo de MHP es proporcionar interoperabilidad entre diferentes aplicaciones y terminales y entre los propios terminales.

DVB-MHP especificó una plataforma estándar basándose en el conocimiento acumulado de experiencias anteriores y tratando de proveer mecanismos que faciliten su adopción en el mercado de la forma menos traumática posible. Para ello, sus principios de funcionamiento se basan en la definición de unos perfiles que marcan la evolución de la plataforma, junto una arquitectura y unos procesos flexibles pensados para facilitar la portabilidad e interoperabilidad de aplicaciones, que están sometidas a un ciclo de vida muy definido.

MHP consta de una serie de estándares que describen completamente el sistema de middleware abierto de DVB. Se define el concepto de “perfil” como un área de aplicación y, como consecuencia, con una serie de capacidades determinadas.

¿Por qué no triunfa MHP?

- La ejecución de la máquina virtual de Java impone la necesidad de unas características en el receptor (procesador, memoria RAM, memoria persistente) que encarece el precio de los receptores interactivos. Por esta razón, la mayoría de los receptores que se vendieron (en particular, en España) no eran interactivos, sino “zappers”.

- Empresas que participaron en la elaboración del estándar comenzaron a reclamar “royalties” por las tecnologías que consiguieron introducir en el estándar y que tienen patentadas. En su momento no reclamaron estos derechos, por lo que se habla de “patentes submarinas”. Algunos expertos consideraron que en estas condiciones, MHP no era viable.

1.2.2. Smart TV

La televisión inteligente [Stv] (traducido del inglés "Smart TV") describe la integración de Internet y de las características Web 2.0 a la televisión digital (en especial, a la televisión 3D) y al set-top box, así como la convergencia tecnológica entre los ordenadores y estos televisores y el STB. Estos dispositivos se centran en los medios interactivos en línea, en la televisión por Internet y en otros servicios como el vídeo a la carta.

Un dispositivo de Smart TV puede hacer referencia a dos conceptos diferentes; por un lado, puede referirse a un televisor que cuenta con la integración de Internet, pero por el otro, también puede hacer referencia a un set-top box para la televisión que ofrece una capacidad de computación más avanzada y una mayor conectividad que un conjunto básico de televisión contemporánea.

Un televisor inteligente permite instalar y ejecutar aplicaciones avanzadas o plugins basados en una plataforma específica, tal como haría el sistema

informático de un ordenador integrado en el televisor o una PC con pantalla "grande". Los televisores inteligentes ejecutan un sistema operativo o el software completo de un sistema operativo móvil, ofreciendo una plataforma para el desarrollador de software.

La televisión inteligente permite al usuario:

-Entregar contenidos de otros equipos o dispositivos de almacenamiento a la red, como fotografías, películas y música utilizando un programa de servicio DLNA,9 como Windows Media Player en el ordenador o NAS, o a través de iTunes.

-Proporcionar acceso a servicios basados en Internet, mediante IPTV [OP1], así como buscar y navegar por Internet, por los servicios de vídeo a la carta, EPG, personalización de contenidos, redes sociales y otras aplicaciones multimedia.

-Visualizar los contenidos en alta definición.

-Lanzar aplicaciones asociadas en un canal concreto, como vídeos relacionados con el contenido, sistemas de votaciones, sistemas de apuestas y participación en concursos, publicidad interactiva.

-Grabar en disco duro interno o externo USB los servicios que se están emitiendo en un momento determinado o copiarlos de Internet.

-Reproducir el contenido de vídeos o música almacenado en un dispositivo USB.

-Instalar aplicaciones sobre la plataforma, como por ejemplo juegos, que se pueden hacer correr en cualquier momento.

-Facilitar las compras realizadas en Internet.

-Controlar de forma remota el televisor con el móvil del usuario, mediante aplicaciones desarrolladas por los dispositivos que cuentan con Android y iPhone.

¿Problema?

El principal problema de las Smart TV reside en que se basa en un modelo cerrado de aplicaciones. Es decir, cada marca crea su propio “estándar” por lo que una misma aplicación tiene que ser creada de diferente forma en función de la marca en la que va a ser ofrecida.

SOLUCIÓN: HBBTV

En este momento aparece la televisión híbrida HbbTV [Hbb], que ofrece esa convergencia entre televisión e internet y además es un estándar abierto.

1.3. HBBTV (Hybrid Broadcast Broadband TV)

1.3.1. ¿Qué es HbbTV?

Hybrid Broadcast Broadband TV o HbbTV [Hbb], es un proyecto paneuropeo de televisión híbrida cuyo objetivo es combinar las emisiones de televisión (broadcast) con servicios de banda ancha (broadband) para entregar al telespectador un servicio de entretenimiento a través de una pantalla de televisión. La Televisión Híbrida trata por tanto de proporcionar un servicio de televisión y de contenido Web mediante banda ancha.

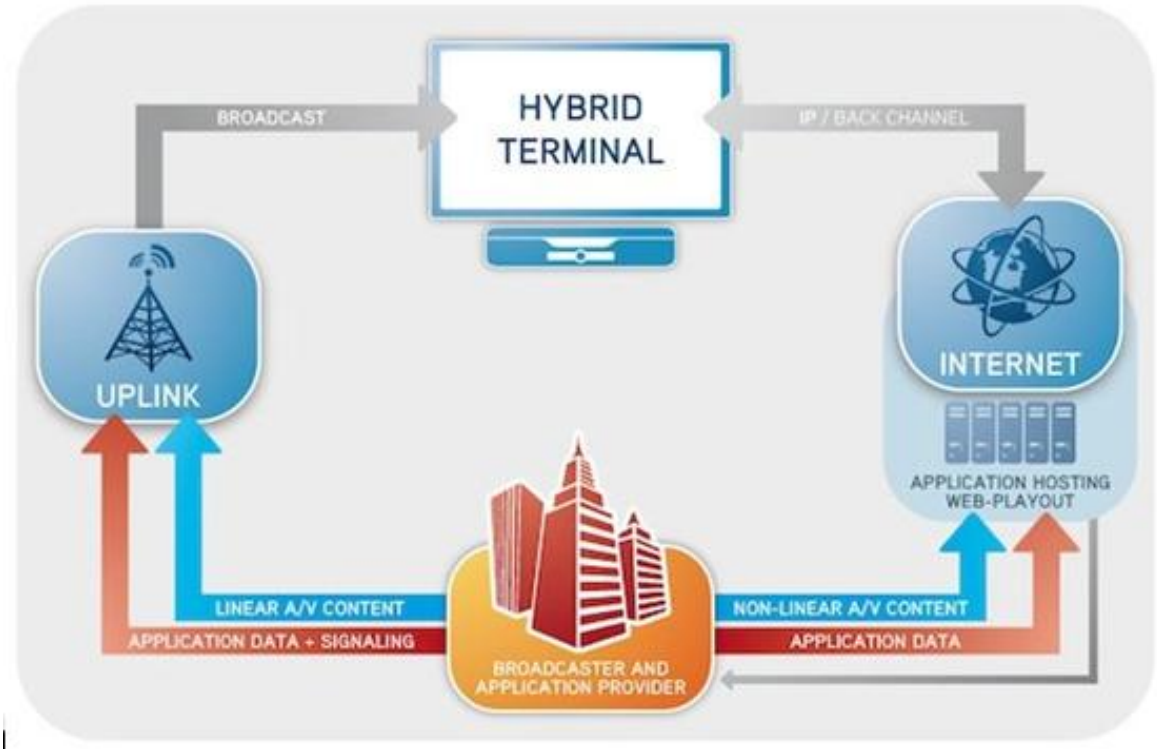


Figura 1.1. Esquema de funcionamiento de HbbTV.

El estándar HbbTV abre la puerta a una experiencia de TV interactiva ya que mediante la adopción de este estándar los telespectadores podrán acceder a nuevos servicios de entretenimiento como:

- Recuperación de programas de televisión Vídeo bajo demanda (VoD).
- Publicidad interactiva.
- Información personalizada en el televisor.

- Votación.

- Juegos.

- Aplicaciones interactivas.

- Navegación Web.

- Redes sociales, etc…

- Servicios relacionados con el propio programa, como son el Teletexto y la Guía Electrónica de Programación (EPG)

Figura 1.2. Servicios prestados por HbbTV.

La HbbTV ofrece una plataforma tecnológica abierta y neutral que combina perfectamente los contenidos de televisión digital (satélite, cable o terrestre) con servicios de banda ancha permitiendo el acceso a servicios de Internet para todos aquellos espectadores que dispongan de un televisor o decodificador con HbbTV.

El objetivo es la realización de transmisiones de contenidos multimedia de calidad estándar (SD) o en alta definición (HD) sobre el estándar híbrido HbbTV [Spe] e IPTV [OP] basado en la calidad y variedad de contenidos, prestando especial atención al establecimiento de una búsqueda sencilla y manejable de estos contenidos.

Consorcio HbbTV

Los miembros fundadores del consorcio HbbTV han desarrollado conjuntamente la especificación HbbTV [Spe] con el fin de crear una norma mundial para los servicios de entretenimiento híbrido. La versión 1.1.1 de esta

especificación ha sido aprobada por la ETSI como ETSI TS 102 796 [ETS], en junio de 2010.

La especificación HbbTV no sólo introduce nuevos componentes técnicos, sino que se basa principalmente en las normas existentes y tecnologías Web como OIPF (Open IPTV Forum) [OIP], CEA [CE], DVB [DBV] y W3C [W3c]. En este sentido, se adapta a las tecnologías disponibles en lugar de implementar un nuevo desarrollo técnico. El estándar proporciona las características y funcionalidades necesarias para prestar servicios de Internet y servicios característicos de radiodifusión. Utilizando la tecnología estándar de Internet permite el desarrollo rápido de aplicaciones. Se definen unos requisitos mínimos que simplifican la implementación de dispositivos, esto limita de inversión necesaria por los fabricantes de la CE para construir dispositivos compatibles.

El consorcio de la HbbTV es una iniciativa paneuropea lanzada en 2009 por el consorcio European Broadcasting Union (Unión Europea de Radiodifusión) que ofrece una alternativa a las tecnologías propietarias y privadas, ofreciendo una plataforma abierta para que los radiodifusores puedan ofrecer sus servicios de valor añadido como VoD al consumidor final.

El grupo de los fundadores del consorcio de la HbbTV estuvo formado por radiodifusores y empresas de la CE, lo que significa que hay un objetivo común de crear servicios y que los radiodifusores puedan ofrecerlos mientras se cumple con las funciones marcadas de los dispositivos de la CE actualmente.

Figura 1.3. Logo del consorcio HbbTV.

1.3.2. HbbTV actualidad

La televisión híbrida ha experimentado una expansión en los últimos años desde su aparición en 2010

1.3.2.1. HbbTV en el mundo

El consorcio HbbTV ha puesto en marcha una campaña de marketing estratégico de cara a fomentar la adopción del estándar HbbTV en todo el mundo.El consorcio HbbTV [Hbb] se estableció oficialmente en 2010 y está compuesto en la actualidad por más de 60 broadcasters y numerosas empresas de electrónica de consumo.

En Europa, tres de cada diez televisores que se venden incluye acceso HbbTv con fuerte tirón en Alemania (con una penetración en venta del 53%), España (26%) y Francia (21%).

La norma ya se ha implementado en varios países europeos, entre ellos Austria, República Checa, Dinamarca, Francia, Alemania, Polonia, España, Suiza, Países Bajos y Turquía, y una gran mayoría de las TV conectados que se vendieron en Europa Occidental implementa ya el estándar HbbTV. Actualmente hay más de dos millones de receptores activos HbbTV en Alemania y más de medio millón de receptores activos en Francia. EL consorcio ha anunciado también nuevas expansiones de la norma en Europa, Australia, Malasia y Vietnam. El estándar HbbTV también está ganando rápidamente partidarios adicionales en todo el mundo, incluyendo América y Asia.

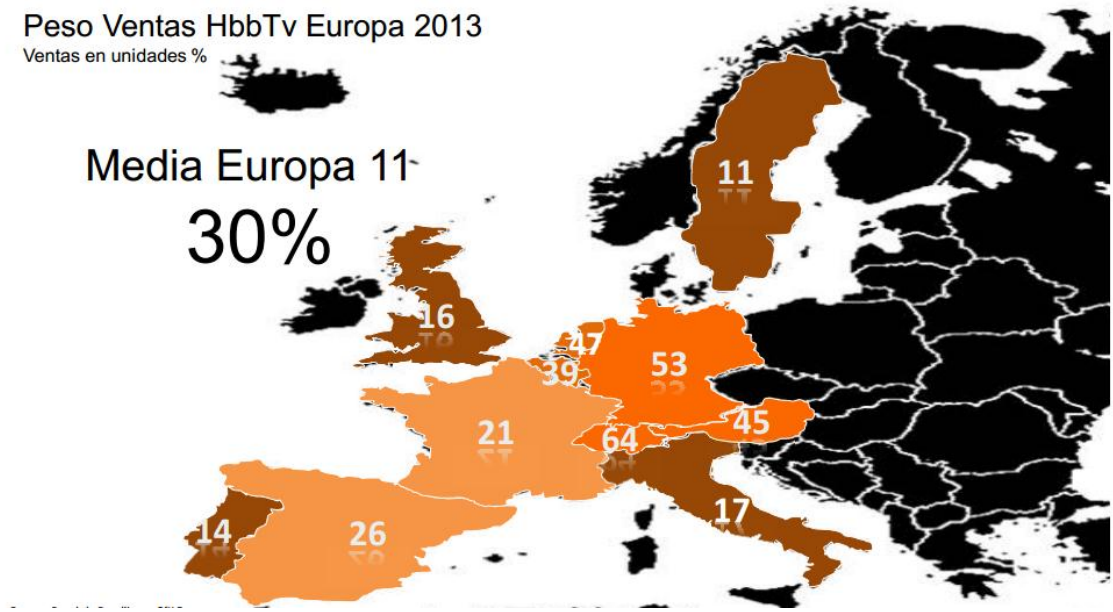


Figura 1.4. Peso de ventas HbbTV en Europa en 2013 [aed].

1.3.2.2. HbbTV en España

Según la Asociación Española de Empresas de Televisión Interactiva (AEDETI) [aed] las perspectivas que el sector tiene para este año son muy buenas y apunta a que 2014 será el año de la "consolidación comercial" de las televisiones conectadas a Internet.

La cantidad de equipos conectados está aumentando exponencialmente en los últimos años y se prevé que en el futuro esto siga creciendo.

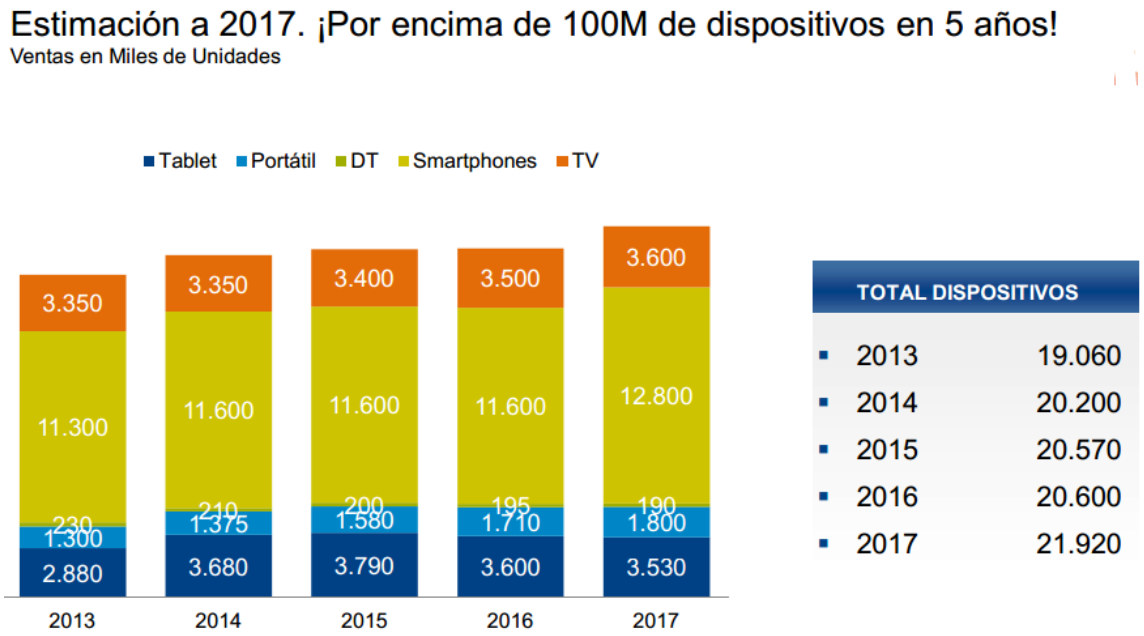


Figura 1.5. Estimación de dispositivos conectados en 2017 [aed].

Actualmente, en España hay 2,7 millones de 'smart TV' o televisores inteligentes, de los cuales aproximadamente 1,6 cuentan con el sistema HbbTV (más del 5% de los hogares españoles), con servicios horizontales de televisión conectada.

Las comunidades en las que más televisores conectados se están vendiendo son Madrid, Cataluña, Comunidad Valenciana, Navarra y Murcia. A la cola encontraríamos Castilla-La Mancha, País Vasco, Rioja y Cantabria.

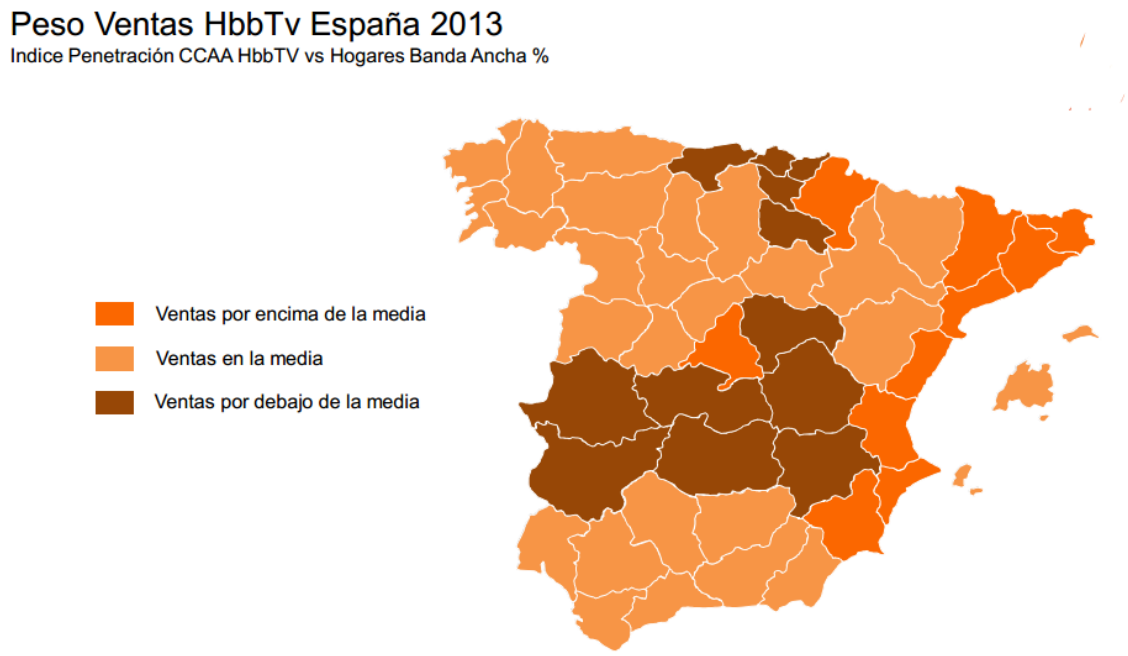


Figura 1.6. Ventas HbbTV en España por comunidades [aed].

En el último año ha habido un incremento de más del 33 por ciento respecto a 2012 en hogares con televisión conectada, liderado por el proyecto de RTVE. La cadena pública ha participado en el lanzamiento comercial de estos servicios con el servicio 'botón rojo' y servicios piloto y comerciales como los realizados con motivo de la Lotería de Navidad.

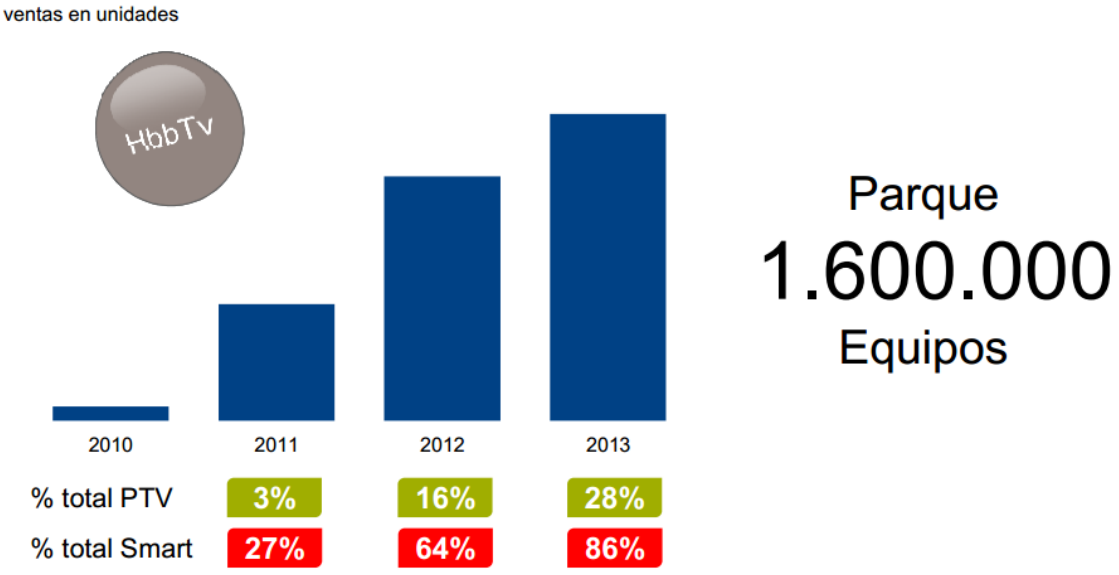


Figura 1.7. Ventas HbbTV en España [aed].

Aunque parece que HbbTV es negativo para los fabricantes (pues cada empresa en su SmartTV puede ofrecer sus propias aplicaciones), los fabricantes han defendido un claro compromiso con el estándar HbbTv. Por

otro lado, han destacado la importancia de la etiqueta TDT híbrida para aclarar al consumidor y armonizar todo el ecosistema.

En cuanto a los desarrolladores de aplicaciones, estos ven en la televisión inteligente una oportunidad de negocio. Sin embargo, han reclamado una mayor armonización de los componentes tecnológicos para involucrar a todos los desarrolladores webs para que pongan en marchan aplicaciones para la televisión inteligente como ya hacen para tablets o smartphones. En este sentido será importante un proceso de certificación de aplicaciones con reglas claras.

1.3.3. Mercado actual

Como ya se ha citado en el apartado anterior, en España hay en la actualidad casi 7 millones de televisiones inteligentes de las cuales 1,6 millones llevan la tecnología HbbTV.

Cuando un usuario se introduce en la página de la empresa *Media-markt* [MM] e introduce hbbtv en el buscador obtiene alrededor de 60 opciones de televisores que incorporan la tecnología HbbTV. Es de destacar que todos estos televisores son de tamaño grande, por lo que adquirir un televisión con HbbTV en estos momentos no es asequible para todo el mundo. Sin embargo, el número de televisores va en aumento por lo que se prevé que un futuro cercano la accesibilidad de estos dispositivos mejorará.

En este caso, se ha elegido la televisión de menor coste encontrada en esta página para realizar un análisis de la misma.

Es un televisor LED de 32” de la marca Panasonic, y el coste actual del mismo es de 299 euros.

Figura 1.8. Televisión con HbbTV incorporada.

Aparece una descripción del televisor, en el que un apartado habla sobre la televisión conectada:

Panasonic integra en este televisor gran variedad de aplicaciones, todas ellas de manejo fácil e intuitivo. Con Mi Perfil Viera podrás personalizar tu perfil de la forma que quieras, accediendo a tu información favorita en cuanto enciendas el televisor.

RTVE, realiza un barrido por las diferentes marcas de televisores. La conclusión global es que para que un televisor sea compatible con HbbTV, tiene que ser del año de 2013, mayor o igual a 32", y pertenecer a la gama alta de televisores (serie 9) [RTVE].

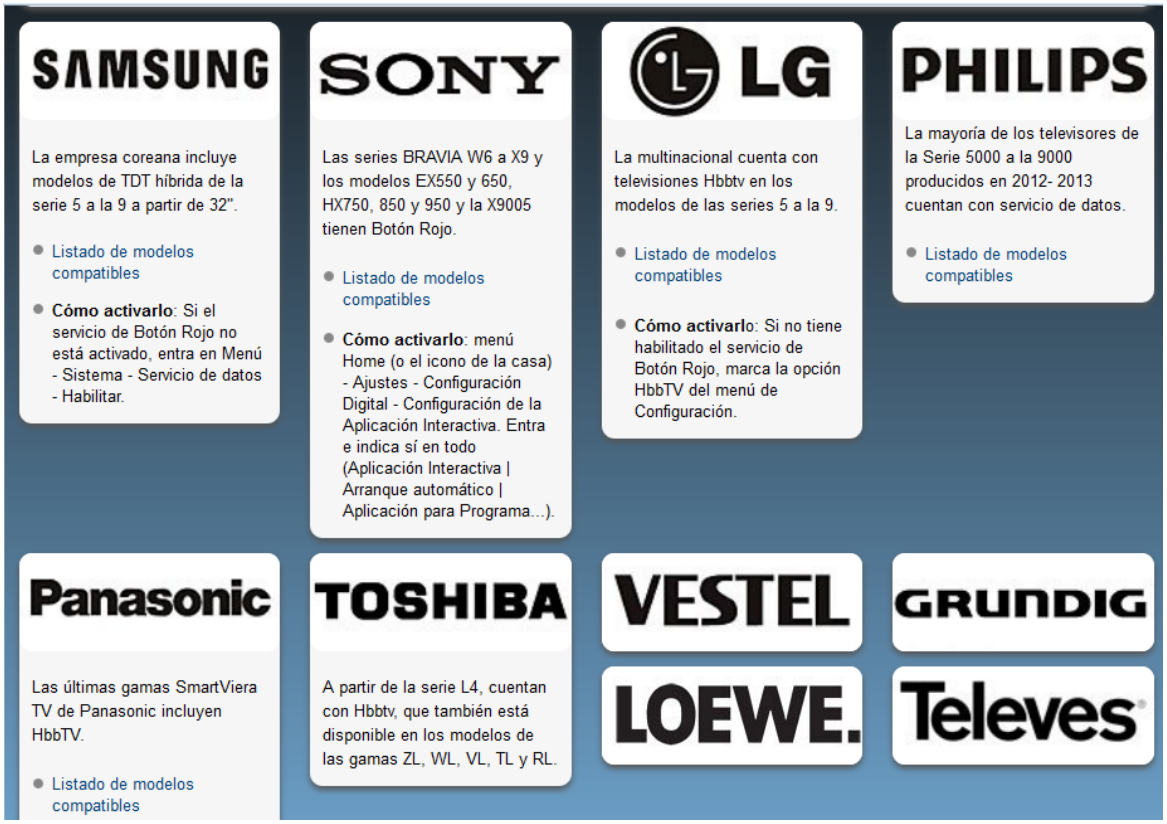


Figura 1.9.Análisis RTVE marcas comerciales y compatibles HbbTV.

Ampliando la buscada a un mercado más amplio, en la página Ebay sólo aparecen 9 resultados al buscar televisores HbbTV lo que demuestra que es una tecnología en expansión. Sin embargo, aparece un televisor de menos coste que el anterior (250), que es un televisor kendo LED de 24 pulgadas e integra tanto Smart TV como HbbTV. Es un televisor alemán, donde la tecnología está más integrada, y por lo que se ve hay televisores de menores dimensiones que en el mercado español actual.

1.3.4. Mit-xperts HbbTV

MIT-Xperts [MIT] es una empresa alemana líder en la televisión interactiva, así como SI / PSI playout / análisis de datos para sistemas basados en DVB. Fundado en 2001, el MIT-Xperts está ampliando continuamente su cartera de aplicaciones, sistemas de emisión, sistemas de autoría, los sistemas de análisis y sistemas de grabación. Siempre trabajan para lograr el mejor valor para el cliente con el aumento de la productividad, reducción de costos y ventajas competitivas.

Esta empresa ofrece una aplicación HbbTV con diferentes opciones. Es un banco de pruebas en el que muestran al consumidor como programar una aplicación HbbTV.

Mediante esta aplicación el usuario puede obtener códigos de programación, para poder así programar en HbbTV, y también posteriormente puede probar sus propias aplicaciones.

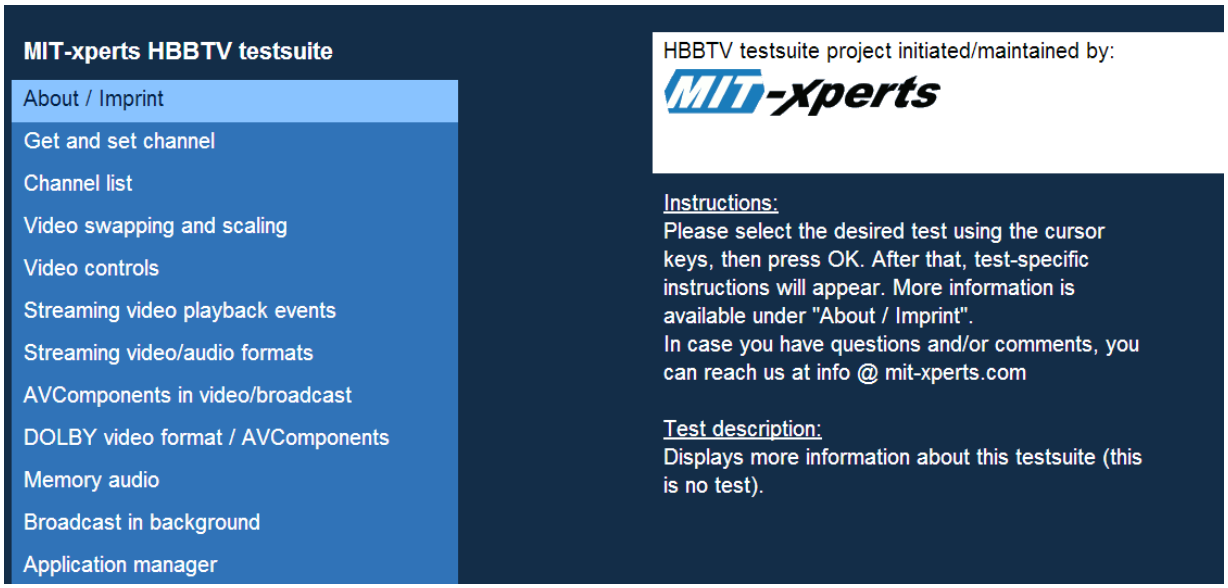


Figura 1.10.mit-Xperts app.

Como se puede ver en la imagen, esta aplicación permite al usuario visualizar diferentes opciones para luego poder aplicarlas en su aplicación. En la siguiente imagen, muestra las diferentes opciones en el ámbito del vídeo, como parar, reproducir, ir a cierto momento del vídeo.

Figura 1.11.mit-Xperts app en apartado de vídeo.

1.3.5. Emuladores HbbTV

Cuando un desarrollador crea una aplicación HbbTV necesita de herramientas para poder probar si el funcionamiento de dicha aplicación es el correcto. Algunos navegadores han desarrollado ciertos paquetes que permiten visualizar aplicaciones ya desarrolladas, y además ofrecen la posibilidad de probar aplicaciones propias.

En este aspecto, el navegador Firefox [Ftv][inf] ofrece un paquete mediante el cual el usuario puede visualizar aplicaciones HbbTV asi como probar sus aplicaciones. A su vez, el navegador Opera [Ope] ofrece otro paquete con las mismas funcionalidades.

Para más información sobre estos emuladores HbbTV consultar el **Anexo I.**

CAPÍTULO 2. TELEVISIÓN CONECTADA

En este primer capítulo se da la base teórica para entender el entorno de la televisión conectada. Por un lado se habla de la red broadcast de televisión digital tal como la entendemos hoy en día, y por otro, se introducen también los conceptos más esenciales de la red broadband de Internet. Finalmente se dedica un apartado a la introducción del estándar HbbTV.

2.1. La red broadcast

2.1.1. Redes de contribución, distribución y difusión de TV digital

La red broadcast es la que se encarga de hacer llegar los contenidos generados los productores audiovisuales y distribuidos por los operadores de radiodifusión a los terminales de una cierta región o zona geográfica. Podemos identificar tres subredes dentro de la red troncal:

- Red de contribución: transporta la señal audiovisual desde su punto de generación u obtención hasta el centro de producción (CP). La señal se procesa en estudios previamente a incorporarse a la emisión.

- Red de distribución: transporta la señal audiovisual desde el centro de producción hasta el centro emisor.

- Red de difusión: transporta la señal audiovisual desde el centro emisor hasta el usuario o consumidor.

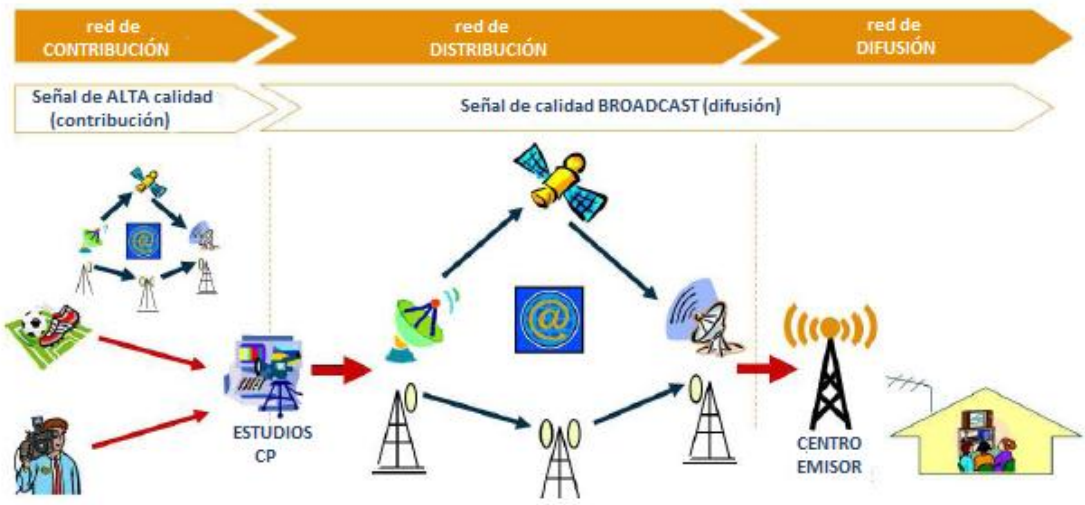


Figura 2.1. Ciclo de señal de radiodifusión.

En la Fig. 2.1 se muestra de manera esquemática el proceso seguido por la señal audiovisual desde su generación u obtención hasta la entrega al usuario.

En la actualidad, las señales de televisión digital se organizan en canales múltiples de 8 MHz de ancho de banda numerados desde el 21 (470-470 MHz) hasta 69 (854-862 MHz). En España los canales disponibles son compartidos por los servicios de difusión públicos y privados de nivel estatal, autonómico o nacional y local.

Una de las características de la televisión digital es que en un solo canal múltiple se puede transportar el contenido de hasta cuatro o cinco programas SD de manera simultánea. Esta es la razón por la que estos canales son llamados "canales múltiples" o "multiplex".

El objetivo de una cabecera digital es el de formar una señal múltiple a partir de las señales entregadas en banda base por el proveedor de contenidos. La señal generada se entrega a la red de difusión, que se encarga de hacerlo llegar al usuario final. Los agentes que participan en el proceso son:

- **CP** (Centro Proveedor de Contenidos): emplazamiento donde se generan los servicios audiovisuales y / o valor añadido (aplicaciones interactivas) que forman parte del múltiplex.

- **CMUX** (Centro de Multiplexación): emplazamiento donde se integran todos los servicios en una única señal: el múltiplex.

- **CE** (Centro Emisor): centro de difusión que transmite la señal múltiplex en su región de cobertura.

La cabecera digital consiste pues en todo el equipamiento instalado en el CP y el CMUX para:

1. Codificar los contenidos que entrega el CPP y generar servicios.

2. Multiplexar todos los servicios en una única señal múltiplex.

3. Adaptar la señal múltiplex para su distribución a los CE que forman la red de difusión.

Los procesos que se llevan a cabo sobre la señal se pueden visualizar esquemáticamente en el dibujo de la Fig. 2.2, donde también se identifica donde tiene lugar cada proceso (en el CP o el CMUX).

Figura 2.2. Procesos aplicados a la señal de radiodifusión.

2.1.2. DVB: Digital Vídeo Broadcasting

Es el estándar de la televisión digital europea (utilizado también en otras partes del mundo), creado en 1993 para dar respuesta a la necesidad de definir un formato común que permitiera la difusión de la Televisión Digital. Algunos de los aspectos que define son:

- Métodos de modulación y corrección de errores relacionados con las transmisiones terrestres (DVB-T/T2), por cable (DVB-C/C2) o por satélite (DVB-S/S2).

- Algoritmo común para los sistemas de encriptación y acceso condicional (DVBC).

- Transmisión de la información de servicio (DVB –SI) que permite al espectador un acceso sencillo y rápido a los servicios o programas que se transportan.

- Utilización del estándar de codificación MPEG-2 [MPE] y H.264 [Cod] para la compresión de las señales audiovisuales.

El consorcio DVB en sí no crea los estándares, sino que proporciona las especificaciones que deben cumplirse a organizaciones como la ETSI, CENELEC, ITU –R , ITU- T o Davic .

En la Fig. 2.3 se muestra un diagrama de flujo donde se representan los diferentes procesos que tienen lugar sobre la señal audiovisual digitalizado y su información asociada.

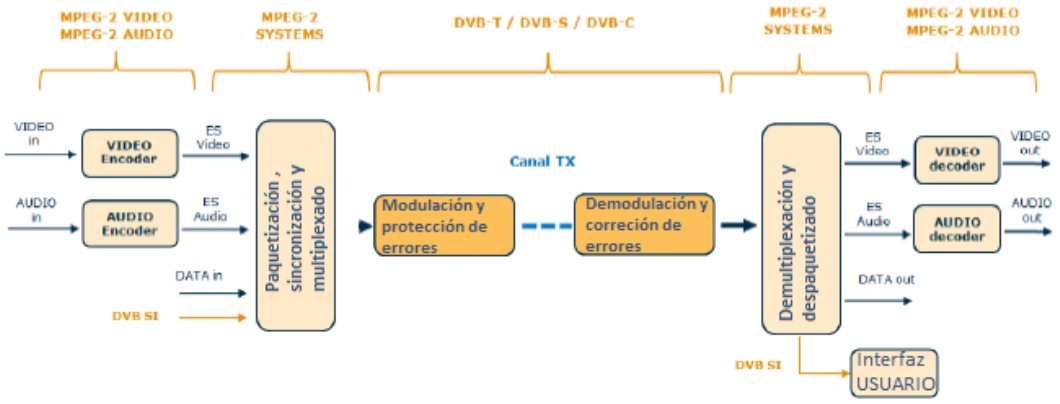


Figura 2.3. Esquema de procesos aplicados a la señal en transmisión y recepción.

2.1.3. Señal audiovisual. MPEG -2 Systems

En caso de que las cámaras de las que se disponga sean analógicas, la señal se digitaliza en los estudios. Se utiliza un sistema compatible con las señales analógicas obtenidas de las cámaras.

- Norma ITU -R 601 / SMPTE 259M: señal SD- SDI (270 Mbps): PAL (625 /50) y NTSC (525/60).

- Norma SMPTE 292M: señal HD- SDI (1.5 Gbps): 1080i y 720p.

- Norma SMPTE 372M: señal HD- SDI (3 Gbps): 1080p.

Se requiere un gran ancho de banda para el transporte de la señal , lo que provoca que haya que comprimirlo siguiendo los estándares MPEG -2 o MPEG -4 Parte 10/H.264 AVC [MPG].

Previamente a la definición de los diferentes formatos de contenedor de MPEG -2 Systems es necesario conocer el concepto de "programa". En televisión digital un programa es una señal audiovisual constituida por un stream de vídeo, uno o más streams de audio y otros streams correspondientes a diferentes flujos de datos (teletexto, datos privados, etc).

Para construir un programa hay que asociar y sincronizar los diferentes flujos de las señales que lo forman (vídeo, audio y datos). La sección "Systems" de MPEG -2, correspondiente al estándar ISO / IEC 13818-1, define dos formatos de contenedor diferentes pero relacionados entre sí:

- **MPEG TS** (Transport Stream): diseñado para transportar señales audiovisuales en canales hostiles (con alta probabilidad de error). No se define con precisión el principio ni el final de la información audiovisual y, por tanto, está orientado a flujos continuos (como es el caso de la radiodifusión de televisión).

- MPEG PS (Program Stream): contenedor diseñado para contenidos con un inicio y un final determinados (películas, vídeoclips, etc.) que almacenan o transmiten en canales menos hostiles (discos duros, discos ópticos, memorias flash o canales con mucha protección contra errores).

La Fig. 2.4 muestra los diferentes flujos de datos y la relación que tienen entre sí.

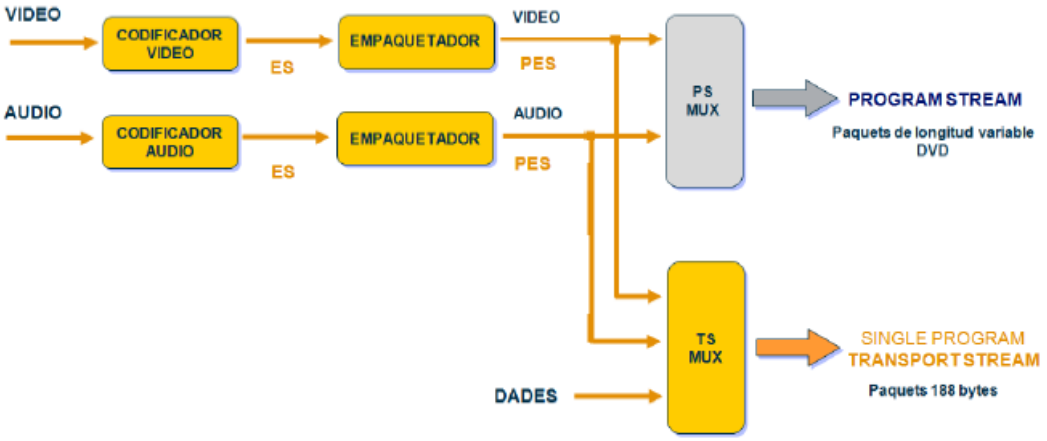


Figura 2.4. Flujos de datos de sistemas MPEG-2, PS y TS.

Donde los acrónimos corresponden a:

- ES (Elementary Stream): flujo continuo de datos que contiene información de una única fuente de vídeo o audio . No contiene información de sincronización relativa a otros flujos, pero sí puede transportar información de ordenación interna de las unidades de acceso (imágenes de vídeo o tramas de audio).

- PES (Packet Elementary Stream): empaquetado de un ES. Los paquetes están formados por una cabecera y el *payload*, que es donde se transporta la información de la ES. Agregar marcas temporales en las unidades de acceso ES para alinear temporalmente y sincronizar los diferentes flujos que forman parte de un programa. Cada PES se identifica con un PID (Packet Identifier) de 13 bits (rango 0 a 8191 en decimal) que permite posteriormente su demultiplexación. Las unidades PES coinciden con las unidades Acceso ES (añadiendo la cabecera PES) y tienen longitud variable (normalmente grande).

- PS (Program Stream): es el resultado de combinar en un solo flujo uno o más PES con una base de tiempo común , a la que están referidos los *timestamps* insertados en la cabecera PES. El flujo PS combinando los paquetes PES de todos los flujos audiovisuales que componen un

programa, de manera que todos avanzan en paralelo y respetan los requerimientos temporales del decodificador para presentar la imagen y el sonido descodificados en los instantes adecuados.

- TS (Transport Stream): resultado de combinar uno o más programas en un solo flujo bajo una base de tiempo común. El flujo TS está compuesto por paquetes de longitud fija (188 bytes: 4 bytes de cabecera y 184 bytes de *payload*).

2.2. La red broadband

Internet nace de un proyecto de investigación financiado por el Ministerio de Defensa norteamericano que tiene como objetivo permitir la compartición de información científica y militar entre ordenadores. Se puso en marcha en 1969 (con el nombre Arpanet) conectando cuatro grandes ordenadores localizados en diferentes universidades del sudoeste los Estados Unidos. En los últimos 40 años ha evolucionado hasta convertirse en una red global de ordenadores que conecta millones de equipos a través de los nodos de conmutación y encaminamiento (routers).

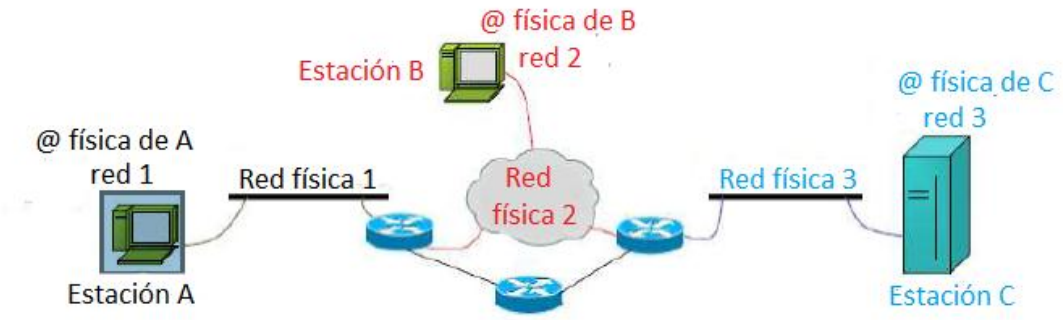


Figura 2.5. Esquema visual de la red broadband.

Las principales características de esta red son las siguientes:

- Permite conectar redes de ordenadores de diferentes arquitecturas.

- Es una red robusta: los paquetes pueden viajar por caminos diferentes en función del estado del enlace en cada instante concreto.

- Es una red descentralizada y distribuida.

- Es escalable: funciona bien independientemente del número de equipos que se conecten, independientemente de la red local a la que pertenezcan.

Debido a su notable complejidad, podemos estructurar Internet en pilas de protocolos siguiendo el modelo facilitado por la pila OSI de la ISO (Fig. 2.6).

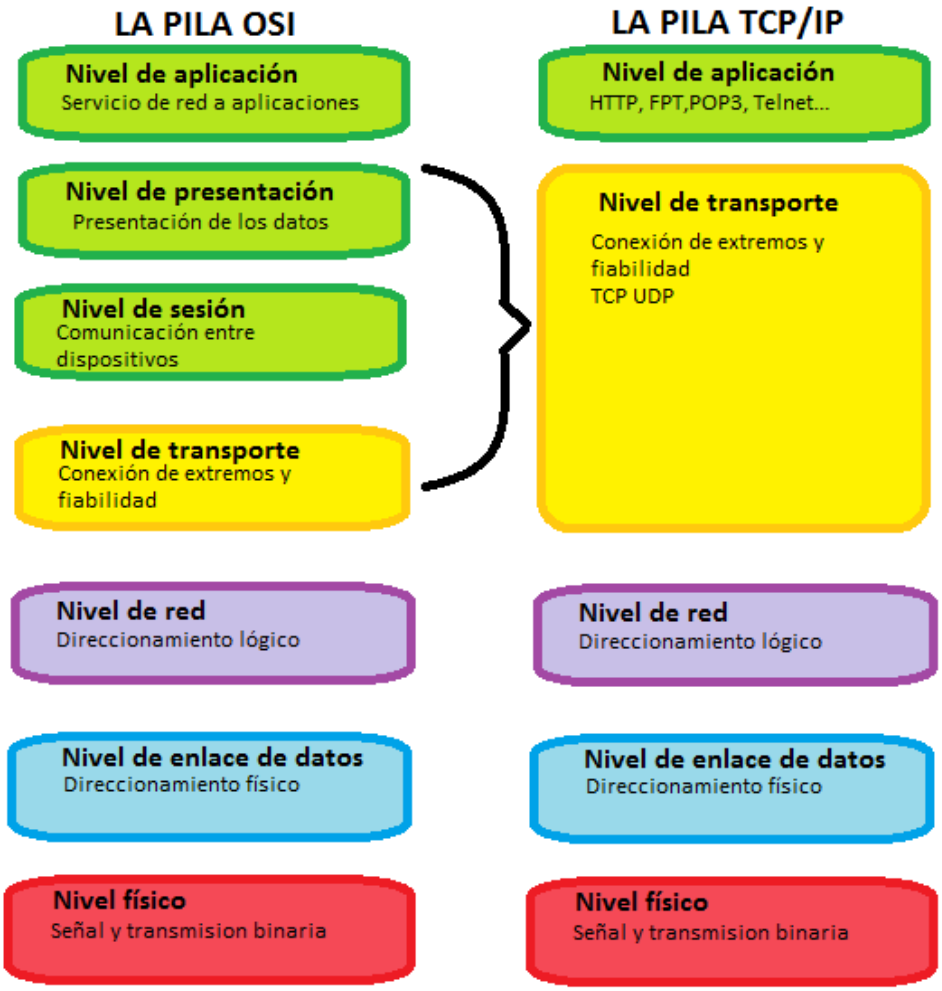


Figura 2.6.Comparación entre la pila OSI de la ISO y la pila TCP/IP.

2.2.1. Los protocolos TCP / IP

Internet Protocol (IP, RFC 791) es el protocolo de red, encargado de la entrega de paquetes [pro]. No está orientado a conexión, lo que significa que cada paquete se trata de forma independiente al resto. No garantiza que el paquete llegue a su destino ni tampoco que lo haga en un tiempo determinado. Transmission Control Protocol (TCP, RFC 793) es el protocolo de transporte más común en Internet junto con UDP (RFC 768). A diferencia del protocolo IP, TCP se ejecuta en los extremos de la comunicación, está orientado a conexión y es fiable. Esta fiabilidad se consigue gracias al mecanismo ARQ, que repite automáticamente los fragmentos de información en función de un diálogo de confirmaciones entre fuente y destino. Este mecanismo de retransmisiones

puede introducir retrasos y provoca que no sea apto para aplicaciones en tiempo real, en las que se suele utilizar el protocolo UDP, que no retransmite los fragmentos y por tanto no es fiable.

2.2.2. El protocolo HTTP

Al nivel de aplicación de la pila TCP / IP encontramos varios protocolos , entre ellos HTTP (RFC 1945 , 2616) , FTP (RFC 959) , POP3 (RFC 1939) , Telnet (RFC 854) , SSH (RFC 4252) , etc .

HyperText Transfer Protocol (HTTP) [pro2] es el protocolo encargado de la transferencia de documentos web. Utiliza los servicios TCP / IP como mecanismo fiable de transferencia de datos. Ofrece una forma de representar peticiones del cliente y respuestas del servidor web. Los tipos más comunes de petición son los siguientes:

- **HEAD**: petición para comprobar el estado de la conectividad cliente - servidor.

- **GET**: petición de un documento.

- **POST**: permite que el cliente facilite información al servidor. En nuestro caso concreto, se han utilizado peticiones POST para el formulario de actualización de la base de datos.

También hay otras peticiones: PUT, DELETE, TRACE, OPTIONS y CONNECT.

Los códigos de respuesta de los servidores se dividen en:

- **1xx Información**: petición recibida, respuesta provisional del servidor.

- **2xx Operación con éxito**: petición recibida y aceptada por el servidor.

- **3xx Redirección**: el cliente debe redirigir la petición.

- **4xx Error de cliente**: petición con errores.

- **5xx Error de servidor**: la petición parece válida pero el servidor no puede atender.

Para identificar recursos en Internet, cada uno de ellos tiene asignada una URL (Uniform Resource Locator) propia. La URL es un identificador formado por el protocolo de comunicación a emplear, la máquina donde se aloja el recurso y el directorio específico dentro de esta máquina. En nuestro caso, la URL de la aplicación HbbTV se indica en uno de los campos de configuración de la tabla AIT.

2.3. El estándar HbbTV (Hybrid broadcast broadband TV)

En este apartado se hace una síntesis de la información sobre el estándar HbbTV disponible en el documento ETSI TS 102 796 v1.1.1 [ETS] que aparece en el año 2010. Esta es la versión que implementaron los dispositivos HbbTV hasta la fecha de julio de 2012. En dicha fecha se publicó la nueva versión 1.5, que complementa la anterior y añade nuevas especificaciones, tales como:

- HTTP adaptive streaming (MPEG DASH) [MPE].

- Esquema de encriptación común para permitir la utilización de múltiples tecnologías DRM.

- Acceso a la información de la tabla EIT3.

En la Fig. 2.7 se muestra el esquema de la situación actual del estándar HbbTV.

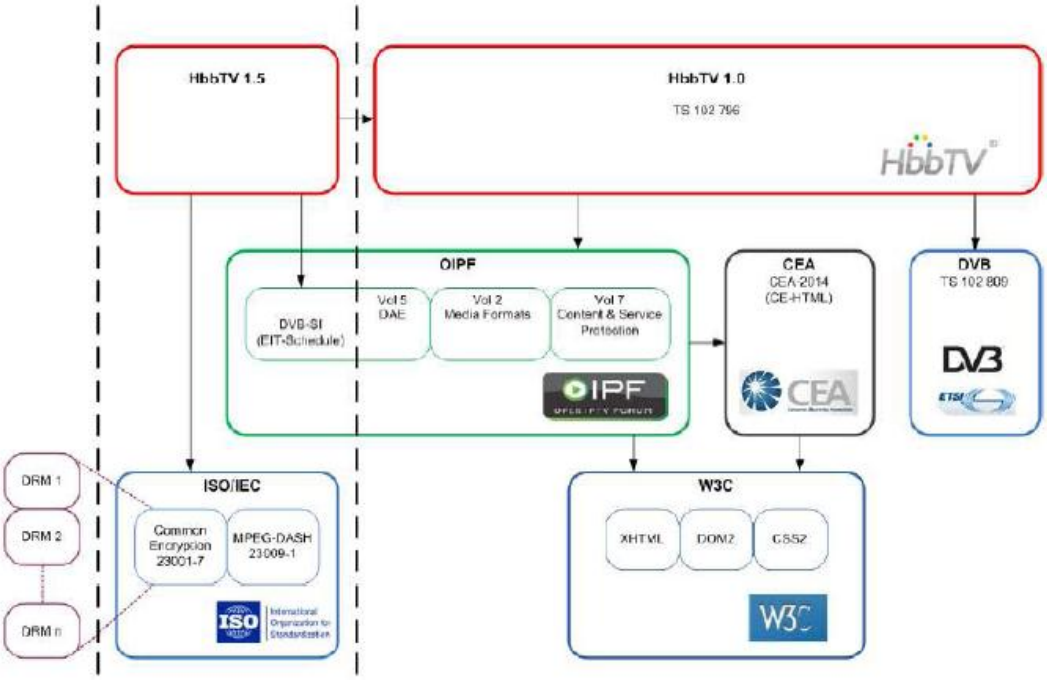


Figura 2.7. Especificación HbbTV.

A continuación desglosaremos la información de la parte correspondiente a la primera versión del estándar.

2.3.1. Introducción

Hybrid Broadcast Broadband TV [Hbb] es una iniciativa de origen europeo que tiene el objetivo de dar un valor añadido a las actuales tecnologías de la televisión digital, poniendo al alcance de los radiodifusores una plataforma

abierta para proporcionar servicios interactivos y contenidos bajo demanda a los consumidores finales. El factor clave se encuentra en la conjunción de dos tecnologías ya existentes: la de la televisión digital (broadcast) e Internet (broadband).

El consorcio industrial encargado de impulsar esta iniciativa incluye radiodifusores terrestres (como RTVE, TV3, Canal +), institutos de investigación, operadores de satélites (SES ASTRA, Eutelsat), fabricantes de terminales (LG, Philips, Panasonic, Sony) y empresas encargadas de producir soluciones software (ANT, Opera, ACCESS, Open TV).

La gran ventaja de disponer de un estándar para esta tecnología híbrida es el hecho de que llegará a todo el que disponga de un terminal que la soporte, independientemente de la marca o el modelo del receptor. De esta manera se consigue un mercado potencial muy amplio. Al tratarse de una plataforma abierta, queda claro que HbbTV se encuentra en una situación privilegiada respecto a otros competidores como pueden ser los mercados cerrados de televisión conectada (IPTV, Samsung Smart TV, etc). Es necesario remarcar que el hecho de que una televisión en concreto tenga acceso a un mercado cerrado de aplicaciones proporcionado por el fabricante, no implica que no pueda acceder a las aplicaciones HbbTV que le llegan señalizadas vía *broadcast*.

2.3.2. Funcionamiento

Como se ha comentado anteriormente, los terminales híbridos que soportan HbbTV tienen la capacidad de conectarse a dos redes de manera simultánea:

- La red broadcast (DVB-T/T2, DVB-S/S2, DVB-C/C2): a través de esta red de televisión digital el terminal puede recibir los contenidos de radiodifusión tal y como los entendemos hoy en día. Estos contenidos reciben el nombre de contenidos audiovisuales lineales (linear A / V contento, en inglés). Además, a través de esta red también se recibe información de la señalización de la aplicación y, en algunos casos, datos de la aplicación.

- **La red broadband** (IP): el terminal se puede conectar a Internet a través de una interfaz broadband (típicamente ADSL). Esto permite establecer una comunicación bidireccional con los proveedores de aplicaciones. Es gracias a esta característica que el estándar HbbTV hace posible la interactividad usuario- aplicación. A través de esta nueva interfaz se reciben datos de la aplicación y contenido audiovisual no lineal (non- linear A / V content, en inglés). En caso de que el terminal no esté conectado a Internet, su funcionamiento es exactamente igual al de los dispositivos receptores de TDT que conocemos.

La Fig. 2.8 es un ejemplo esquemático donde se muestra un terminal híbrido conectado en la red de radiodifusión DVB-S y en Internet.

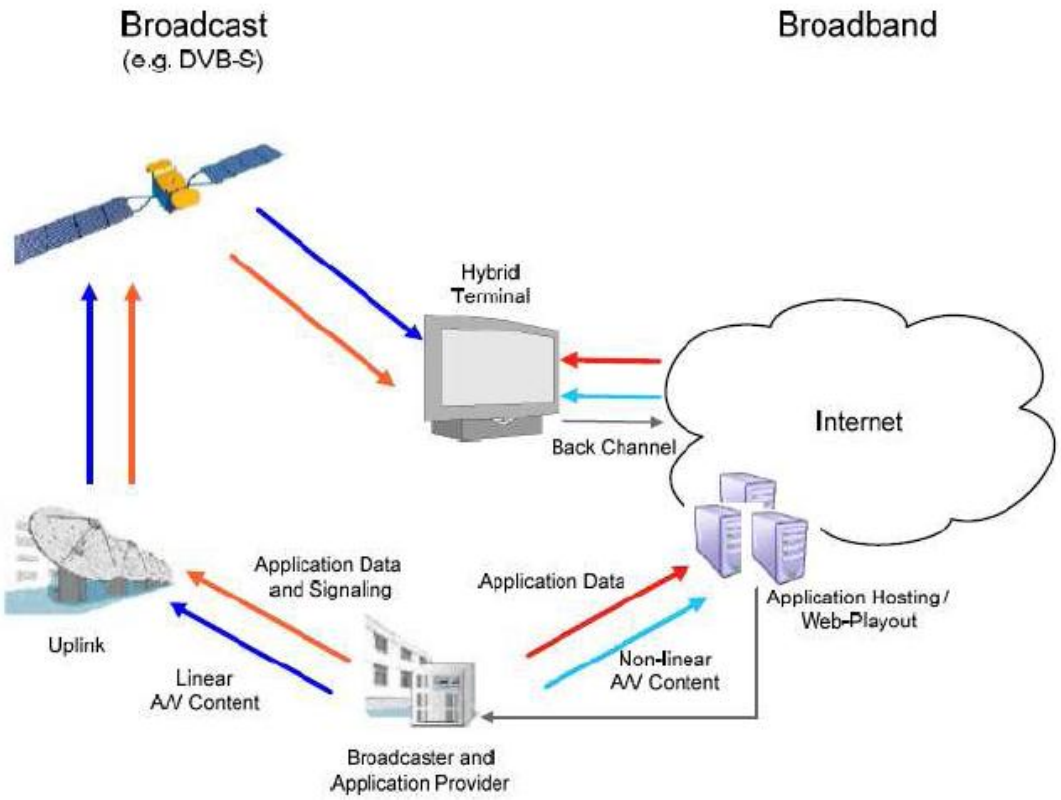


Figura 2.8. Esquema de una red híbrida.

2.3.3. Tipos de aplicaciones

Los servicios accesibles gracias a HbbTV serán facilitados por los radiodifusores, los propios fabricantes de los dispositivos u otros proveedores independientes de aplicaciones. Se puede intuir entonces que según quien proporcione el servicio, este tendrá unas características diferentes. La diferenciación principal es desde donde se lanza la aplicación (desde un portal, desde un canal en concreto, desde diversos canales, etc).

Según define el estándar, las aplicaciones se dividen en dos categorías:

- Aplicaciones **broadcast - related**: se declaran a la señalización DVB y, tal como se indica en su nombre, son aplicaciones relacionadas con un servicio de radiodifusión. Por tanto, este tipo de aplicación se lanza desde uno o más canales de televisión. El contenido puede llegar vía broadcast (parcialmente insertado en el stream) o vía broadband (más comúnmente). Algunos ejemplos de aplicaciones *broadcast related* son:

  - Aplicaciones de botón rojo.

  - Servicios relacionados a un programa.

- Teletexto digital (puede ser lanzado con el botón "txt" del mando).

Las aplicaciones de botón rojo o los servicios relacionados a un programa suelen señalizarse como AUTOSTART, de manera que su ejecución comienza en cuanto se sintoniza el canal al que están asociadas. Sin embargo, se suele dejar a la elección del usuario el hecho de acceder o no a los contenidos no lineales con un mensaje previo de aceptación.

- Aplicaciones **broadcast - independent**: se trata de aplicaciones que sólo están disponibles vía broadband y que no se declaran en ningún servicio del TS. Los proveedores de estas aplicaciones pueden ser fabricantes u otras empresas independientes. Algunos ejemplos de este tipo de aplicación son:

  · Juegos online a través de la TV

  · Redes sociales, compartición de fotografías, etc.

  En este caso, el escenario más común a partir del cual se lanzan las aplicaciones es un portal disponible mediante algún botón del mando a distancia.

Esta separación en la organización de las aplicaciones, junto con el hecho que las aplicaciones broadcast -related deben ser declaradas en la señalización DVB, asegura a los radiodifusores que en su canal sólo se presentarán sus aplicaciones.

2.3.4. Escenarios

Tal como se ha explicado, las aplicaciones se pueden declarar en la señalización, pueden no estar declaradas o bien se pueden referenciar en un portal independiente. Al mismo tiempo, la navegación entre diferentes aplicaciones es posible, es decir, es posible ejecutar una aplicación desde otra. A modo de ejemplo, algunas veces, una aplicación broadcast -related puede contener un enlace a una aplicación broacast - independiente para ampliar el abanico de contenidos y/o servicios disponibles.

El consumidor sintoniza diferentes canales de televisión, cada uno de los cuales puede tener una aplicación broadcast -related asociada (o no). En caso afirmativo, la aplicación se puede lanzar automáticamente (modo AUTOSTART) al sintonizar el canal (por ejemplo, mostrando un gancho o hook). Es el usuario pues quien decide si acceder o no a los contenidos interactivos disponibles. Para entrar a la aplicación tan sólo hay que pulsar el botón rojo del mando a distancia una vez aparece la opción en la pantalla.

Una aplicación broadcast -related puede redirigir al usuario hacia otra aplicación broadcast-related o hacia una broadcast-independent. Este último tipo de aplicaciones no pueden ser lanzadas directamente mediante señalización asociada a un canal. Sólo se puede acceder si se proviene de otra aplicación o bien a través de un portal independiente, típicamente diseñado por el fabricante. Las características del portal, así como la navegación una vez se ha accedido, quedan fuera del estándar HbbTV ya que dependen única y exclusivamente del fabricante.

2.3.5. Tecnologías utilizadas en HbbTV

La especificación HbbTV está basada en estándares ya existentes. Así pues, no se trata de un nuevo desarrollo técnico, sino más bien de una conjunción específica de tecnologías ya disponibles. En la Fig. 2.9 se muestra el resumen de las tecnologías a partir de las cuales se desarrolla la especificación, y que describiremos a continuación.

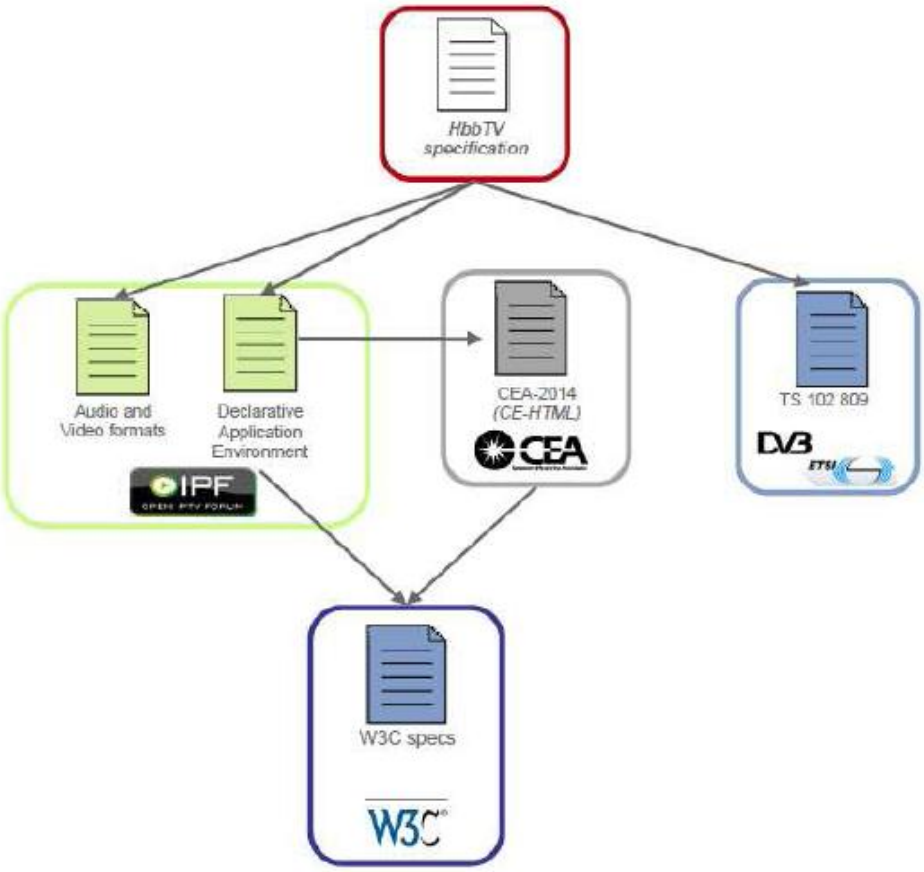


Figura 2.9. Tecnologías utilizadas en el estándar HbbTV.

-CEA - 2014 - Web-based Protocol and Framework for Remote User Interface (Web4CE), también conocido como **CE-HTML** [CE] define las funcionalidades esenciales del navegador. Está basado en los estándares web del W3C y especifica un perfil HTML para los

dispositivos. Utiliza XHTML 1.0, DOM 2, CSS y JavaScript. También contiene otros elementos importantes como por ejemplo la definición de los códigos los mandos a distancia de las televisiones.

Algunos dispositivos soportan XmlHttpRequest (conocido como AJAX), que permite actualizaciones dinámicas de los datos de las aplicaciones sin la necesidad de recargar la totalidad de los datos.

- Open IPTV Forum Release [OIP]: esta especificación se ha desarrollado específicamente para sistemas IPTV basados en el estándar DVB , pero las APIs (JavaScript) que proporciona pueden ser utilizadas en cualquier sistema híbrido DVB , como por ejemplo HbbTV. Estas APIs, entre otras funciones, permiten combinar la imagen de la emisión en directo (broadcast) de la TV con las páginas HTML que definen la aplicación. Para obtener más información sobre la API utilizada en este proyecto, consultar **ANEXO II,** facilitado por un alumno que realizó un estudio previo de esta materia.

- ETSI TS 102 809 [TS] "Signalling and carriage of interactive applications and services in Hybrid Broadcast Broadband environments": define la señalización de las aplicaciones HbbTV sobre Transport Stream para la su difusión a través de los estándares DVB. Estas aplicaciones se pueden ejecutar en el contexto de un servicio específico en un múltiplex DVB. Esto se hace a través del Application Information Table (AIT) en el servicio DVB correspondiente y se indica al Program Map Table (PMT).

Para la implementación de la aplicación se han utilizado los lenguajes especificados en el documento CEA- 2014 que son los indicados por el estándar HbbTV de la ETSI en su punto 4.4. [ETS].

HTML

Siglas de HyperText Markup Language [htm]. Tal como indica su nombre, se trata de un lenguaje de marcado, no de programación. Es el lenguaje predominante para la elaboración de páginas web.

Su función principal es la de describir el contenido y la estructura del texto de la aplicación, así como la adición de imágenes. Se define una estructura de contenidos muy esencial, no es útil para diseñar la apariencia gráfica. Sin embargo, permite aplicar algunos estilos gráficos sencillos.

El lenguaje HTML se escribe utilizando etiquetas o tags, que permiten diferenciar los diferentes elementos y hacer referencia de manera individual o colectiva al código donde se definen los estilos gráficos de la

aplicación. La estructura de un elemento se compone de la etiqueta inicial, el contenido y la etiqueta final o de cierre. La etiqueta inicial puede contener atributos para caracterizar el elemento. A los atributos es donde se sitúan los identificadores para diseñar posteriormente el grafismo. HTML es un lenguaje estático, es decir, define los contenidos de manera fija y el navegador es capaz de interpretar el código y mostrar los resultados.

PHP

Lenguaje de programación utilizado para generar páginas web y / o aplicaciones dinámicas. No se ejecuta el cliente, sino que el lenguaje PHP [php] es interpretado por el servidor, que a su vez devuelve la respuesta en formato HTML, de manera que su presencia es transparente al navegador cliente. PHP pues, ofrece el dinamismo no presente en HTML, ya que permite obtener los contenidos cambiantes a partir de llamadas a una base de datos, por ejemplo.

El código PHP se puede incluir dentro del código HTML entre las etiquetas *< ? Php* y *? >*. Todo el contenido encerrado en estas etiquetas será traducido por servidor previamente a su envío al cliente conjuntamente con el resto de código HTML.

CSS

Siglas de Cascade Style Sheet [css]. Es el lenguaje para describir los estilos gráficos o la presentación de los elementos definidos en el código HTML.

Aunque los estilos se pueden definir en la etiqueta de apertura de cada elemento mediante el atributo style, es más sencillo y más práctico a la hora de realizar el mantenimiento de la aplicación definir en un documento CSS separado. Además, de esta manera, se puede definir un estilo común para un conjunto de elementos identificados con el mismo atributo class.

La inserción de un documento CSS a un HTML se hace mediante la inclusión de la siguiente sentencia en la cabecera:

```
<link rel = "stylesheet" type = "text/css" href = "doc.css">
< / link>
```

JavaScript (incluyendo AJAX)

Lenguaje de programación utilizado para crear páginas web o, en este caso, aplicaciones HbbTV dinámicas que se ejecuta en el cliente. Como aplicación dinámica entendemos la ejecución de diferentes acciones al

generar diferentes tipos de eventos, permitiendo la interactividad usuario aplicación. Otra de las características importantes de JavaScript [jav], y que como veremos más adelante será de utilidad para el desarrollo de la aplicación, es la posibilidad que ofrece de crear, leer y eliminar cookies. Las cookies sirven para almacenar información en su navegador cliente con el objetivo de recuperar más adelante y poder discernir entre varias acciones en función de los datos obtenidos.

La inserción de código JavaScript en un documento HTML se puede hacer de dos maneras: por una parte, dentro del cuerpo del código HTML podemos insertar el código JavaScript entre las etiquetas *<script> < / script>*. Por otro lado, podemos añadir un documento JavaScript externo incluyendo la siguiente sentencia en la cabecera HTML:

<script type="text/javascript" src="doc.js"> < / script>

Las siglas AJAX provienen de Asynchronous JavaScript and XML. Esta tecnología permite utilizar el objeto XMLHttpRequest para intercambiar datos de manera asíncrona con el servidor web mediante peticiones HTTP y HTTPS. La ventaja principal que ofrece es la capacidad de actualizar determinadas partes de la aplicación sin necesidad de recargarla por completo.

SQL

El lenguaje de consulta estructurado (SQL) [sql] es un lenguaje de base de datos normalizado, utilizado por los diferentes motores de bases de datos para realizar determinadas operaciones sobre los datos o sobre la estructura de los mismos. Pero como sucede con cualquier sistema de normalización hay excepciones para casi todo; de hecho, cada motor de bases de datos tiene sus peculiaridades y lo hace diferente de otro motor, por lo tanto, el lenguaje SQL normalizado (ANSI) no nos servirá para resolver todos los problemas, aunque si se puede asegurar que cualquier sentencia escrita en ANSI será interpretable por cualquier motor de datos.

2.3.6. Aplicaciones broadcast -related

Las aplicaciones relacionadas a un canal broadcast se pueden encontrar en uno de los siguientes tres estados cuando se inicia su ejecución en modo AUTOSTART:

1. Mostrando el gancho o hook de acceso.
2. No mostrando ninguna interfaz de usuario (sólo la emisión broadcast).
3. Mostrando la interfaz de usuario completa.

En general, en las aplicaciones que van asociadas a servicios de TV, el estado inicial más común es el primero, de manera que el usuario es informado de la disponibilidad de contenidos extra mediante un breve mensaje. El resto de partes de la aplicación no se muestran hasta que no se pulsa el botón rojo del control remoto. En cuanto a las aplicaciones asociadas a un servicio de radio, lo más lógico es que el estado inicial sea el tercero, ya que la ejecución y la aparición automática de todos los elementos de la aplicación no supone ningún inconveniente en un proceso de comunicación exclusivamente auditivo.

Cuando se produce un salto desde los estados 1 ó 3 hacia el segundo, la aplicación se ha encargarse de:

- Eliminar todos los elementos gráficos de la pantalla.

- Parar la reproducción de vídeo o audio proveniente del canal broadband.

- Reanudar la emisión del servicio broadcast (si es que este se ha detenido previamente).

- Reescalar el vídeo a pantalla completa (si es que se ha escalado previamente).

- Activar el audio del broadcast (si se ha desactivado con anterioridad).

- Dejar de escuchar cualquier tipo de evento que no sea pulsar el botón rojo para volver otra vez al estado 3.

Cuando una aplicación cambia desde el estado 2 al 1 o al 3, deberán producirse las siguientes acciones:

- Mostrar los gráficos y los contenidos de la aplicación actualizados.

- Informar al terminal cuáles son los eventos a escuchar en este nuevo estado.

De acuerdo con las definiciones técnicas del ciclo de vida de una aplicación que se pueden consultar en el punto 6 del documento ETSI TS 102 796 v1.1.1 [ETS], las aplicaciones deben parar cuando se lanza una nueva o cuando se produce un cambio de canal. Además, se pueden parar por sí solas como resultado de una acción del usuario (pulsar el botón rojo) o debido a su lógica interna.

2.3.7. La tabla AIT

Según se especifica en el punto 5.3.2 del documento ETSI TS 102 809 [TS], el stream principal de la tabla PMT de un servicio DVB que tenga una o más aplicaciones HbbTV asociadas debe hacer referencia a otros streams que permitan:

- Localización del stream que transporta la AIT (Application Information Table).

- Localización del / stream (s) que transportan datos de la aplicación (si se da el caso).

La tabla AIT proporciona la información necesaria para que el receptor pueda localizar y ejecutar las aplicaciones relacionadas a un servicio de radiodifusión.

2.3.8. Formato de los contenidos

El estándar no define los requisitos de los formatos de vídeo y audio del canal broadcast. Estos requisitos están definidos en las especificaciones correspondientes cada uno de los mercados donde se desarrollan los dispositivos. En cuanto a los formatos del resto de contenidos, los encontramos definidos en la especificación OIPF Media Formatos [OP3].

Tabla 2.1. Formatos de vídeo y audio broadband

System Format	Video Format	Audio Format	MIME Type
TS	AVC_SD_25 AVC_HD_25	HEAAC E-AC3	video/mpeg
MP4	AVC_SD_25 AVC_HD_25	HEAAC E-AC3	video/mp4

Como se puede ver son codecs de la familia MPEG-4, incluyendo H.264/AVC, de vídeo de definición estándar (720x576 píxeles) y de alta definición (1920x1080) a 25 imágenes/s, y codificadores de audio 5.1 del tipo Dolby AC3 y High-Efficiency Advanced Audio Coding.

Tabla 2.2. Formatos de los contenidos de audio puro.

Audio Format	MIME Type
MPEG1_L3	audio/mpeg
HEAAC	audio/mp4

En este caso también hay que destacar la aceptación del formato mp3.

Tabla 2.3. Formatos de las imágenes estáticas.

Image Format	MIME Type
JPEG	image/jpeg
GIF	image/gif
PNG	image/png

2.3.9. Protocolos de red

Todos los protocolos soportados por HbbTV son los que se definen a la especificación OIPF Protocolos v1.1. [OP4]. A continuación haremos un breve resumen:

2.3.9.1 Protocolos para streaming

El estándar soporta el protocolo HTTP 1.1 para streaming unicast. El terminal ha de poder almacenar datos por adelantado equivalentes a un máximo de 10 segundos de reproducción. Para los contenidos de vídeo MPEG4/AVC y audio MPEG / AAC es opcional que los terminales soporten streaming unicast utilizando RTSP y RTP.

2.3.9.2 Protocolos de descarga

Se utiliza el protocolo HTTP tanto en modo de descarga persistente como de descarga progresiva, en la que el contenido se puede empezar a visualizar previamente a la finalización de descarga.

2.3.9.3 Protocolos de transporte

El protocolo HTTP tal como se define en RFC 2.616 y el HTTP sobre TLS definido en los RFC 2818 y RFC 5246 (transporte seguro, encriptado y autentificado) deben ser soportados en el transporte de las aplicaciones para la red broadband.

2.3.9.4 Cabecera HTTP User -Agent

Todas las peticiones HTTP hechas por una aplicación HbbTV deben incluir una cabecera User -Agent con una estructura tal y como se describe a continuación:

HbbTV/1.1.1 (<capabilities> ; [<vendorName>] ; [<modelName>] ;[<softwareVersion>] ; [<hardwareVersion>] ; <reserved>)

La definición de cada uno de los campos que forman la cabecera puede encontrarse en el punto 7.3.2.4 del documento ETSI TS 102 796 v1.1.1.

CAPÍTULO 3- Iniciación en tecnología HbbTV

Una vez conocida la televisión híbrida, y estudiado su estándar, se pasó a visualizar aplicaciones, estudiar código ajeno y realizar pequeños cambios en aplicaciones ya existentes.

Finalmente, se creó una pequeña aplicación como primera prueba de aplicación HbbTV

3.1. Lenguajes de programación

Para programar en HbbTV es necesario conocer ciertos lenguajes básicos de programación. En este caso, con conocer JavaScript [jav] y HTML [htm] + CSS [css] es suficiente para poder crear pequeñas aplicaciones. Pero en caso de querer realizar aplicaciones de mayor entidad y con mayor dinamismo es preciso dominar también el lenguaje PHP [php].

3.1.1. Javascript

Para realizar el estudio de dicho lenguaje, se utilizó un tutorial encontrado en la web [tu1].

El tutorial está pensado para que pueda ser desarrollado por una persona que no conoce lenguajes de programación, es decir "JavaScript mi primer lenguaje". El objetivo de este sitio es poder aprender JavaScript en forma sencilla viendo un concepto teórico, luego algunos ejercicios resueltos y por último y lo más importante, efectuar una serie de ejercicios. Puede resolver los ejercicios en el sitio, probarlos y ver los resultados. Se recomienda primero ver el detalle del tema, pasar posteriormente a la ejecución de problemas ya resueltos del tema tratado (podemos hacer modificaciones sobre dicho problema) y finalmente resolver los ejercicios propuestos.

Orden del concepto	Concepto	Detalle del concepto	Problema resuelto	Problema a resolver
1	Conceptos de Algoritmo, Programa y Lenguaje de Programación.			
2	Qué es JavaScript?			
3	Variables.			
4	Entrada de datos por teclado.			
5	Estructuras secuenciales de programación.			
6	Estructuras condicionales simples.			
7	Estructuras condicionales compuestas.			
8	Estructuras condicionales anidadas.			
9	Operadores lógicos && (y) en las estructuras condicionales.			
10	Operadores lógicos || (o) en las estructuras condicionales.			

Página: 1 2 3 4 5 6 7 8 9

Figura 3.1. Menú del tutorial JavaScript.

Dicho tutorial contaba con 100 temas, y cada uno de esos temas proponía un problema a resolver. En el cd se pueden encontrar dichos problemas resueltos.

3.1.2. HTML+CSS

En este caso, se realizó un tutorial mediante vídeos [tu3]. Era una serie de veinte vídeos en los cuales, empezando desde cero, se iban explicando las diferentes nociones básicas de estos lenguajes. En cada uno de los vídeos iban apareciendo diferentes ejemplos que se fueron realizando al finalizar cada vídeo.

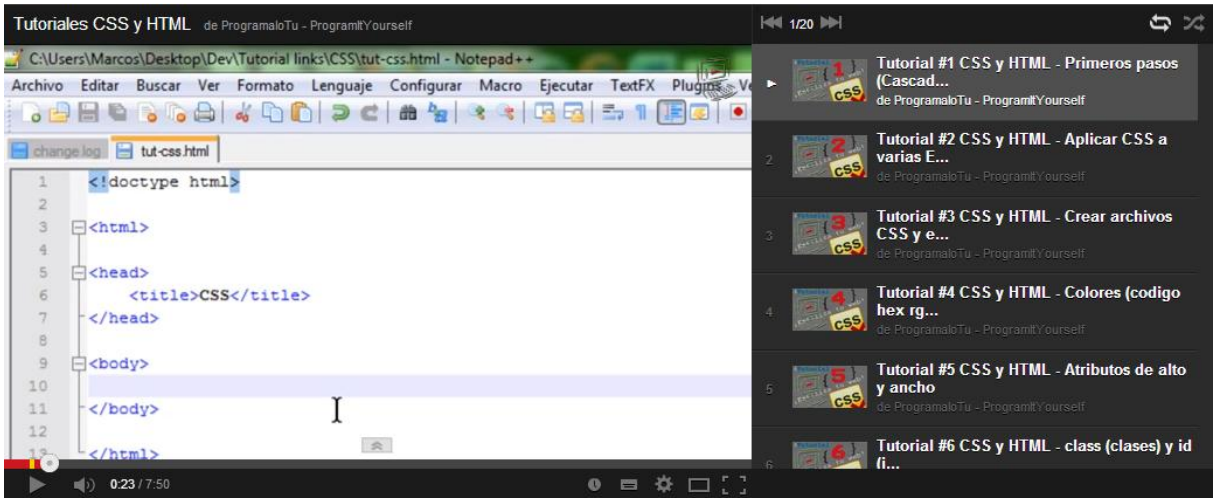


Figura 3.2.Imagen del entorno de trabajo en tutorial HTML+CSS.

3.1.3. PHP

En este caso, se usó de nuevo un canal de vídeos para conocer de forma básica dicho lenguaje de programación [tu2].

Figura 3.3.Imagen del entorno de trabajo en tutorial PHP.

En este caso, dicho tutorial contaba con un total de veintidós vídeos, y cada uno de ellos proponía una actividad a resolver una vez visualizado el mismo. Se completaron todas las actividades propuestas.

3.1.4. MySQL & PHPMyAdmin

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

Se utilizó ***PHPMyAdmin*** para manejar la administración de MySQL a través de páginas web, utilizando Internet. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 62 idiomas.

Para aprender a manejar PHPMyAdmin, se realizó un tutorial mediante contenido audiovisual. Dicho tutorial contaba con una serie de vídeos que iban explicando desde cero el funcionamiento de esta herramienta, con diferentes ejemplos que se fueron realizando simultáneamente al tutorial. Dichos ejemplos se pueden encontrar en el CD, en la carpeta llamada pruebas Database.

Figura 3.4.Imagen de entorno de trabajo en tutorial phpMyAdmin.

Una vez estudiados, de forma básica, todos los lenguajes de programación implicados en HbbTV, el siguiente paso fue el de estudio de código ajeno.

3.2. Estudio de código ajeno

Antes de comenzar a programar, es necesario conocer las estructuras básicas de programación de un estándar. En este caso, y partiendo de la aplicación ofrecida por mit-Xperts [MIT], se comenzó a estudiar los diferentes archivos que aparecían en dicha aplicación.

Básicamente, aparecen archivos de tres tipos de extensión. Extensión **.css**, que son las hojas de estilos, extensión **.php**, en donde se definen los objetos principales y direcciones o donde reside el código del programa principal y por último los archivos con extensión **.js** en los que se encuentran definidas las diferentes funciones que posteriormente son utilizadas en la aplicación.

3.2.1. CSS

Lógicamente, en el archivo con extensión**.css** se encontraban las hojas de estilos. Las hojas de estilo en cascada hacen referencia a un lenguaje de hojas de estilos usado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas. Su aplicación más común es dar estilo a páginas webs escritas en lenguaje HTML y XHTML. Por lo tanto en este archivo se encontraba definido el aspecto que tenía la aplicación de mit-xperts en lo relacionado a colores, tipos y tamaños de letra, formatos de menú (en este caso menú vertical…).

```
ul.menu {
  padding: 0;
  margin: 0;
  position: absolute;
  width: 400px;
}
.menu li {
  display: none;
  overflow: hidden;
  width: 400px;
  height: 34px;
  text-align: left;
  padding-left: 11px;
  padding-top: 5px;
  font: 20px sans-serif;
  color: #ffffff;
  background-color: #A4A4A4;
}
.menu .lisel {
  color: #000000;
  background-color: #ffffff;
}
```

Figura 3.5.Ejemplo de definición de menú.

3.2.2. PHP

En el archivo **.php** se encontraron diferentes funciones. Como ya se ha dicho PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

En este archivo se encontraban definidas diferentes funciones. La primera de ellas definía el tipo de contenido que presentaba la aplicación de mit-Xperts.

```
function sendContentType() {
  header('Pragma: no-cache');
  header('Cache-Control: no-cache');
  header('Content-Style-Type: text/css');
  $uagent = strtolower($_SERVER['HTTP_USER_AGENT']);
  if (strstr($uagent, 'firefox') || strstr($uagent, 'chrome')) {
    header('Content-Type: application/xhtml+xml; charset=UTF-8');
  } else {
    header('Content-Type: application/vnd.hbbtv.xhtml+xml; charset=UTF-8');
  }
}
```

Figura 3.6.Función que define contenidos de aplicación.

En la siguiente función estudiada, se podía ver cómo mit-Xperts definía la definir la versión HbbTV, la API utilizada, y las raíces entre archivos utilizados en la aplicación. Era aquí donde se definían pues los saltos entre las diferentes páginas que posee la aplicación estudiada.

```
function openDocument($title=TITLE, $allscripts=1, $addheaders='') {
  global $ROOTDIR;
  echo '<?xml version="1.0" encoding="utf-8" ?>'."\n";
  echo '<!DOCTYPE html PUBLIC "-//HbbTV//1.1.1//EN" "http://www.hbbtv.org/dtd/HbbTV-1.1.1.dtd">'."\n";
  echo "<html xmlns=\"http://www.w3.org/1999/xhtml\" xml:lang=\"en\" lang=\"en\">\n";
  echo "<head>\n";
  echo "<title>$title</title>\n".$addheaders;
  echo "<meta http-equiv=\"content-type\" content=\"Content-Type: application/vnd.hbbtv.xhtml+xml; charset=UTF-8\" />\n";
  echo "<link rel=\"stylesheet\" type=\"text/css\" href=\"$ROOTDIR/base.css\" />\n";
  echo '<script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>';
  if ($allscripts) {
    echo "<script type=\"text/javascript\" src=\"$ROOTDIR/keycodes.js\"></script>\n";
    echo "<script type=\"text/javascript\" src=\"$ROOTDIR/base.js\"></script>\n";
  }
```

Figura 3.7.Función con versión, API, relación con otros archivos.

A continuación aparecía una función en la que se definía un objeto. Con esta función lo que se define es la gestión de la aplicación, memoria, estado, permisos, etc. Y a su vez también se define la interfaz de usuario de configuración y la información de configuración. La configuración de hardware del OITF se gestiona a través de una instancia del objeto LocalSystem. Este

proporciona acceso a información sobre el hardware y proporciona un punto de entrada para configurar las salidas y las interfaces de red de la OIF.

```
function appmgrObject() {
  if ($_REQUEST['demo']) return '';
  return '<object id="appmgr" type="application/oipfApplicationManager" style="position: absolute;
}
```

Figura 3.8. Función de gestión de aplicación.

Por último aparecía una función en la que se definía un objeto de vídeo que posteriormente utiliza mit-Xperts en su aplicación. Define también características de dicho objeto como tamaño y posición.

```
function videoObject($left=0, $top=0, $width=1280, $height=720) {
  if ($_REQUEST['demo']) {
    global $ROOTDIR;
    return '<img id="video" style="position: absolute; left: '.$left.'px; top: '.$top.'px; width: '.$width.'px; height: '.$height.'px;"
  }
  return '<object id="video" type="video/broadcast" style="position: absolute; left: '.$left.'px; top: '.$top.'px; width: '.$width.'px;
```

Figura 3.9. Función de definición de objeto vídeo.

3.2.3. JavaScript

A continuación se estudiaron los archivos con extensión .**js** . El primero de ellos, era una librería HbbTV que hace una equivalencia entre las teclas del ordenador y los botones del control remoto.

```
if (typeof(KeyEvent.VK_0)!='undefined') {
   var VK_0 = KeyEvent.VK_0;
   var VK_1 = KeyEvent.VK_1;
   var VK_2 = KeyEvent.VK_2;
   var VK_3 = KeyEvent.VK_3;
   var VK_4 = KeyEvent.VK_4;
   var VK_5 = KeyEvent.VK_5;
   var VK_6 = KeyEvent.VK_6;
   var VK_7 = KeyEvent.VK_7;
   var VK_8 = KeyEvent.VK_8;
   var VK_9 = KeyEvent.VK_9;
}
```

```
if (typeof(VK_0)=='undefined') {
   var VK_0 = 0x30;
   var VK_1 = 0x31;
   var VK_2 = 0x32;
   var VK_3 = 0x33;
   var VK_4 = 0x34;
   var VK_5 = 0x35;
   var VK_6 = 0x36;
   var VK_7 = 0x37;
   var VK_8 = 0x38;
   var VK_9 = 0x39;
}
```

Figura 3.10. Equivalencia entre teclas del ordenador y botones del control remoto.

El siguiente archivo con extensión **.js,** se encargaba de definir funciones globales que se podían utilizar en todas las páginas que tenía la aplicación. No son las únicas funciones definidas pues luego ser verá como en el archivo

index.php, se definen otras funciones que únicamente serán utilizadas en dicha página.

En estas funciones globales que definía mit-Xperts aparecían funciones del tipo de inicio o fin de aplicación, navegación en el menú, iniciación de vídeo, detección de eventos de teclado…

```
function initApp() {
  try {
    var app = document.getElementById('appmgr').getOwnerApplication(document);
    app.show();
    app.activate();
  } catch (e) {
    // ignore
  }
  setKeyset(0x1+0x2+0x4+0x8+0x10);
}
```

Figura 3.11. Ejemplo de función global.

3.2.4. index.php

Por último, se estudió el archivo llamado ***index.php.*** En este archivo, se encontraron 3 lenguajes diferentes de programación. En primer lugar, aparecían una serie de sentencias en php. En ellas, se llamaban a los archivos requeridos por este archivo, y se definían el tipo de servidor y llamadas a algunas de las funciones de las comentadas anteriormente como la de definición del tipo de contendido.

```
<?php
$ROOTDIR='.';
require("$ROOTDIR/base.php");
$referer = $_SERVER['HTTP_REFERER'];
$i = strrpos($referer, '/');
$referer = substr($referer, 0, $i);
$referer = substr(strrchr($referer, '/'), 1);
$referer = addcslashes($referer, "\0..\37'\\");

sendContentType();
openDocument();
?>
```

Figura 3.12. Sentencias PHP en archivo index.php.

A continuación, y mediante lenguaje javascript, llamaban a algunas de las funciones globales antes definidas y además añadían funciones locales para este para esta página en concreto. Desde estas sentencias se iniciaba la aplicación.

```
window.onload = function() {
  menuInit();
  registerKeyEventListener();
  setDescr();
  initApp();
  nameselect('<?php echo $referer; ?>');
```

Figura 3.13. Llamada a funciones globales.

```
function nameselect(snam) {
  if (!snam) return;
  for (var i=0; i<opts.length; i++) {
    var check = opts[i].getAttribute('name');
    if (check==snam) {
      menuSelect(i);
      setDescr();
      break;
    }
  }
}
```

Figura 3.14. Funciones locales definidas en javascript.

Por último, aparecía programado en HTML, la forma de los diferentes objetos, la situación de los mismos, los colores de la página, la definición de las partes del menú… En definitiva, aparecía definido el aspecto que se encontraba posteriormente el visitante de la aplicación.

```
<div style="left: 0px; top: 0px; width: 1280px;
 height: 720px; background-color: #A4A4A4;" />
```

Figura 3.15. HTML definiendo color de fondo de la página.

3.3. Modificaciones en código ajeno

Una vez estudiado el código utilizado por mit-Xperts [MIT] en su aplicación, era el momento de realizar pequeñas modificaciones para comprobar que se habían entendido los conceptos y que se era capaz de realizar cambios con conocimiento en aplicaciones ya construidas.

En primer lugar, y mediante el editor de notas Notepad ++, se abrió el archivo de hojas de estilo y se trataron de hacer modificaciones en lo referente a cambios en color de fondo, cambios en tipos de fuente, en definición de diferentes menús…

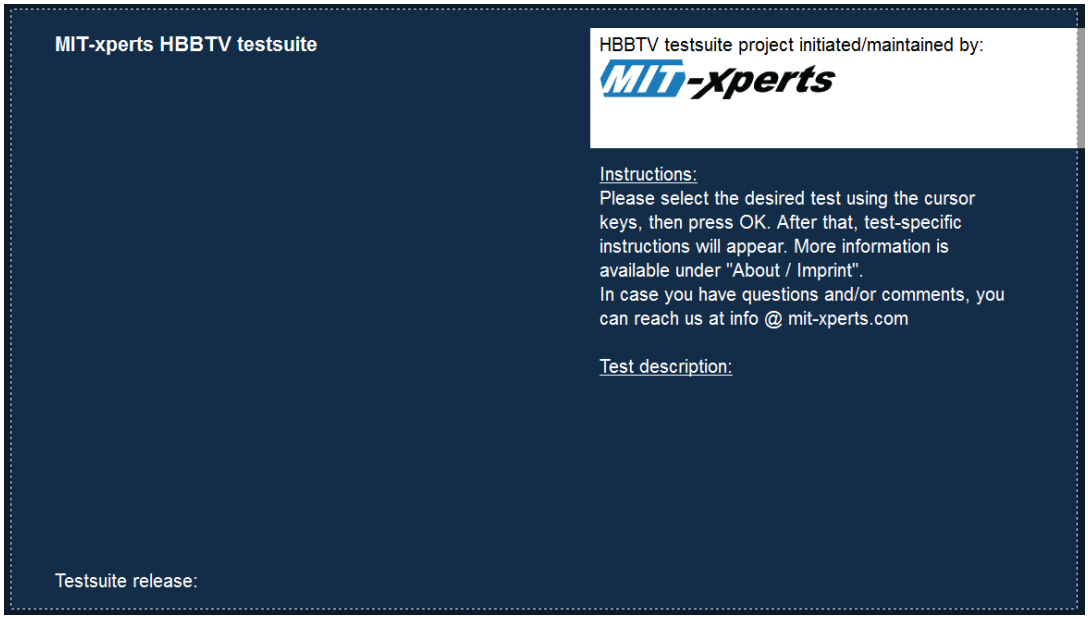


Figura 3.16. Apariencia habitual Mit-xperts app.

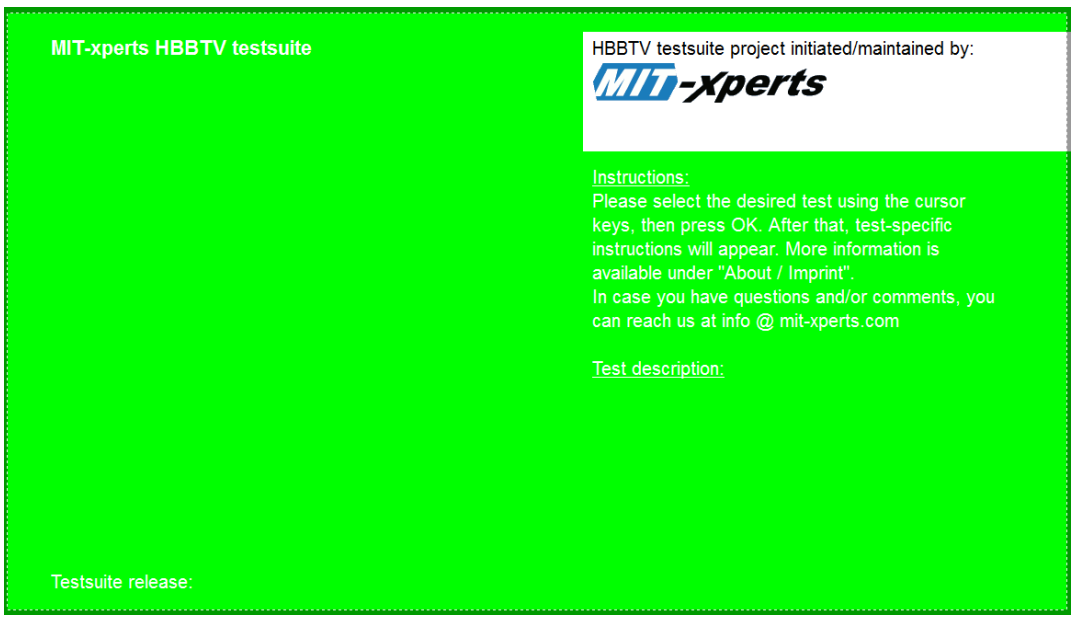


Figura 3.17. Modificación del color de fondo en Mit-xperts app.

Como ya sabíamos, el lenguaje CSS sirve para organizar la presentación y aspecto de una página web, y en este caso lo hace para una aplicación HbbTV. Con el tutorial realizado previamente, se vio que CSS es muy intuitivo y sencillo una vez se llega a aprender, ya que para su definición siempre se hace uso de un identificador de etiqueta HTML (como por ejemplo <H1>), y posteriormente se indica con qué aspecto se quieren mostrar todas las etiquetas <H1> que aparecen en un documento.

En lo referente al archivo php, se realizaron cambios en lo referente a la navegación entre diferentes páginas. Se cambiaron las selecciones del menú, cambiando las raíces de las direcciones.

Los principales cambios se realizaron en el programa principal. En él, se realizaron cambios de tamaños, de posiciones de objetos, en definitiva de aspecto. También se realizaron cambios en algunas de las funciones como por ejemplo en la función de navegación en el menú o en la de cerrar la aplicación.

3.4. Primera aplicación

Una vez estudiado el estándar de televisión híbrida, después de visualizar diferentes aplicaciones ya creadas, de estudiar el código de una de esas aplicaciones (mit-Xperts) y de realizar cambios sobre el código de dicha aplicación, era el momento de comenzar a realizar las primeras pruebas propias.

3.4.1. Objetivo

El objetivo principal de estas primeras pruebas, era ni más ni menos ,que el de tener una toma de contacto con el mundo de la programación en HbbTV. Para ello se trataron de combinar los códigos disponibles de otras aplicaciones y los conocimientos aprendidos en los cursos de lenguajes de programación.

Con la primera aplicación construida, de nombre ***mipruebadeapp***, lo que se trató fue de realizar pequeñas pruebas, como pueden ser aspectos de página, o cambios de textos.

Uno de los requisitos de esta primera aplicación fue que tenía que tener una página con un vídeo, lo que obligaba a estudiar las especificaciones del estándar y de esta forma conocer los formatos de vídeo que cumplen dicha normativa y la forma de codificarlos

Como se ve en la imagen siguiente el esquema de dicha aplicación fue muy sencillo, pues la profundidad máxima era de solo un salto.

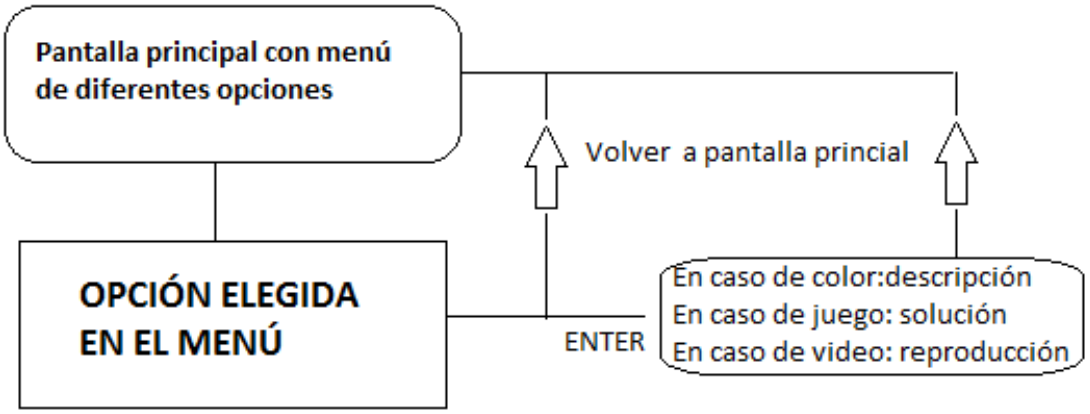


Figura 3.18. Esquema navegación en mipruebadeapp.

Para la creación de dicha aplicación, se usará el mismo patrón de archivos que utiliza mit-Xperts para el desarrollo de la suya:

-base.php
-base.css
-base.js
-settings.php (librería de botones)
-index.php

3.4.2. Primera aplicación: mipruebadeapp

Para esta primera aplicación, se creó una pantalla principal, en la cual aparecía un menú de selección y una imagen con la carta de colores. En dicho menú aparecen una serie de opciones las cuales se dividen en 3 tipos. Los tres tipos son:

-color (de este tipo hay 8 selecciones posibles)

-juego

-vídeo

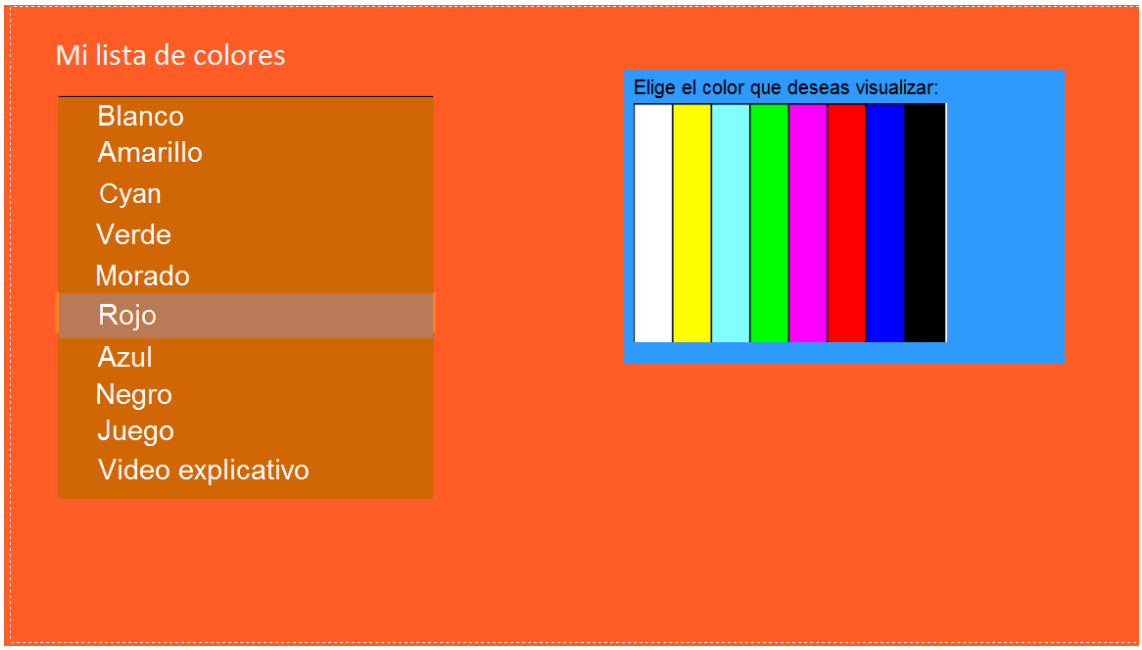


Figura 3.19. Pantalla inicial de la aplicación.

Como se ve en la imagen, la pantalla de inicio consta de un título, de una imagen con la carta de colores y por último con un menú en el cual se ve la selección actual de diferente color para que el usuario sepa en qué opción del menú se encuentra.

```
body {
    background-color: #d83101;
    height: 1280px;
    margin: 0px;
    padding: 0px;
    width: 720px;
}
```

Figura 3.20. Definición en css del fondo de la aplicación.

En la siguiente imagen se puede ver el código del menú realizado, donde se define la posición, el tamaño, el color…El apartado de *menú.lisel* es en el que definimos que la selección actual tenga un color diferente al resto de selecciones posibles.

```
ul.menu {
    padding: 0;
    margin: 0;
    position: absolute;
    width: 400px;
}
.menu li {
    display: none;
    overflow: hidden;
    width: 400px;
    height: 34px;
    text-align: left;
    padding-left: 11px;
    padding-top: 5px;
    font: 20px sans-serif;
    color: #ffffff;
    background-color: #ff5000;
}
.menu .lisel {
    color: #ff0000;
    background-color: #89c3ff;
}
```

Figura 3.21. Definición del menú.

Tanto la navegación a través del menú como la selección elegida por el usuario se definen en una función javascript que se encuentra en el programa principal.

```
function handleKeyCode(kc) {
  if (kc==VK_UP) {
    menuSelect(selected-1);
    setDescr();
    return true;
  } else if (kc==VK_DOWN) {
    menuSelect(selected+1);
    setDescr();
    return true;
  } else if (kc==VK_LEFT){
    menuSelect(selected-9);
    setDescr();
    return true;
  } else if (kc==VK_RIGHT){
    menuSelect(selected+9);
    setDescr();
    return true;

  } else if (kc==VK_ENTER) {
    var liid = opts[selected].getAttribute('name');
    if (liid=='exit') {
      closeApp();
    } else {
      document.location.href = liid+'/';
    }
```

Figura 3.22. Función javascript.

Por último, el cuadro en el que se encuentra la imagen con la carta de colores y con el breve texto son definidos en la parte de HTML del código del programa principal así como los nombres que aparecen en el menú.

```
<div style="left: 690px; top: 100px; width: 450px; height: 300px; background-color: #2E9AFE;"
  <div class="txtdiv" style="left: 10px; top: 4px;
  width: 400px; height: 30px; color: #000000;">Elige el color que deseas visualizar:</div>
  <div class="imgdiv" style="left: 10px; top: 34px;
  width: 450px; height: 250px; background-image: url(colores.png);"></div>
</div>

<ul id="menu" class="menu" style="left: 100px; top: 100px;">
  <li name="blanco" >blanco</li>
  <li name="amarillo" >amarillo</li>
  <li name="cyan">cyan</li>
  <li name="verde" >verde</li>
  <li name="magenta" >magenta</li>
  <li name="rojo" >rojo</li>
  <li name="azul" >azul</li>
  <li name="negro" >negro</li>
  <li name="juego" >juego</li>
  <li name="video" >video explicativo colores</li>
</ul>
```

Figura 3.23.Código HTML.

Si la selección elegida por el usuario es uno de los 8 colores, la pantalla que se encontrará el usuario será de la siguiente forma.

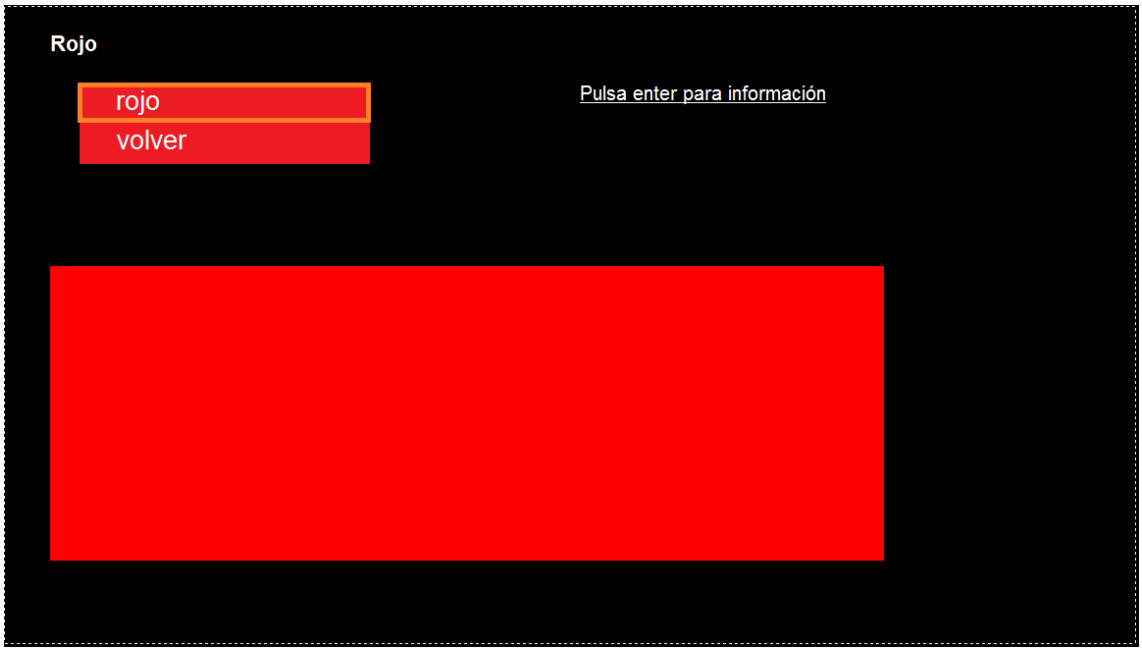


Figura 3.24. Selección color rojo.

Como se ve en la imagen, aparece un pequeño menú en el que solo hay dos opciones, la del color y la de volver a la pantalla de inicio. A su lado hay una frase que si se selecciona el color en el menú cambia por una descripción de

dicho color y lo que este significa. Por último aparece un cuadro del color seleccionado.

```
function runStep(name) {
  if (name=='contrib') {
    document.getElementById('txtdiv').innerHTML = "El color rojo es el del fuego y el de la sangre,
    por lo que se le asocia al peligro, la guerra, la energía, la fortaleza, la determinación,
    así como a la pasión, al deseo y al amor.";

    menuSelect(opts.length-1);
  }
}
```

Figura 3.25. Función javascript.

Esta función javascript es la que se encarga de que si el usuario pulsa sobre el color en el menú, desaparezca la frase y en su lugar aparezca una descripción del significado del color rojo.

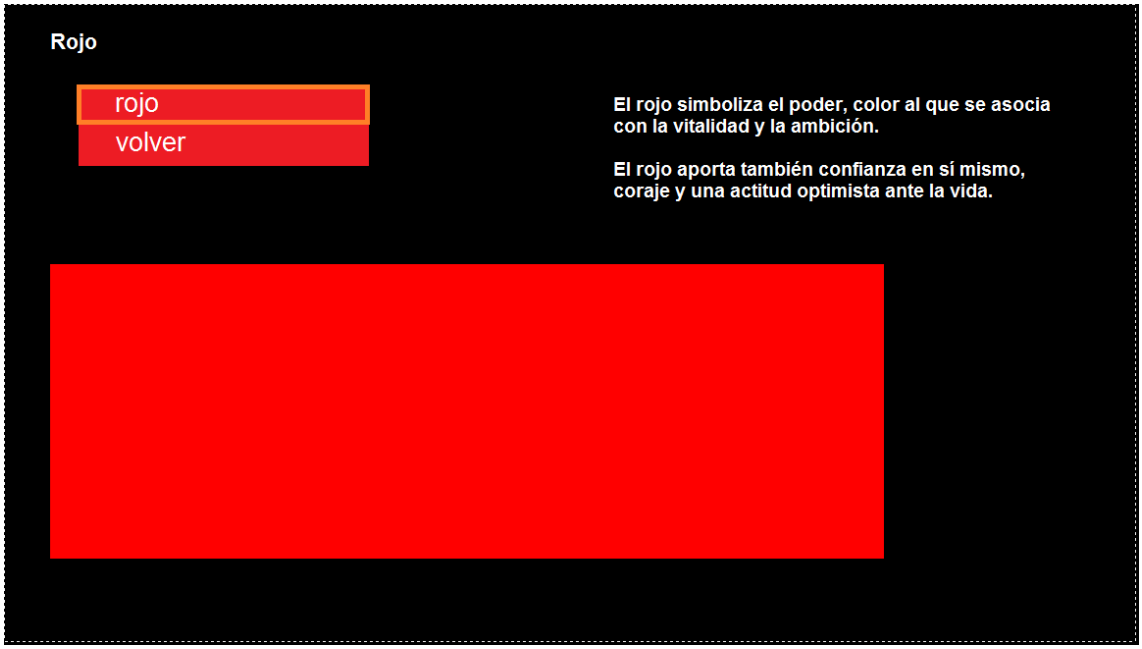


Figura 3.26. Pantalla con descripción color rojo.

Con cualquiera de los otros siete colores restantes, el resultado sería el mismo pero cambiando el cuadro de color y la descripción del significado, pues cada uno tiene su propia descripción como es lógico.

En cambio, si el usuario pulsa sobre la opción juego, se encontrará con una pregunta y con un menú de opciones para responder a dicha pregunta. Si el usuario responde de forma correcta el programa actuará de una forma y si lo hace de forma incorrecta el programa responderá de otra.

Figura 3.27. Pantalla inicial del juego.

En este caso, lo que aparece en el programa principal es una imagen con la carta de colores, un menú similar al de la pantalla de inicio, y un cuadro de texto con la pregunta que el usuario tiene que contestar.

```
function runStep(name) {
  if (name=='azul') {
    document.getElementById('txtdiv').innerHTML = "intentalo de nuevo";

    menuSelect(opts.length-1);
  }else if (name=='rojo') {
    document.getElementById('txtdiv').innerHTML = "intentalo de nuevo";

    menuSelect(opts.length-1);
      }else if (name=='verde') {
    document.getElementById('txtdiv').innerHTML ="intentalo de nuevo;

    menuSelect(opts.length-1);
      }else if (name=='negro') {
    document.getElementById('txtdiv').innerHTML = "intentalo de nuevo";

    menuSelect(opts.length-1);
      }else if (name=='amarillo') {
    document.getElementById('txtdiv').innerHTML = "intentalo de nuevo";

    menuSelect(opts.length-1);
      }else if (name=='cyan') {
    document.getElementById('txtdiv').innerHTML = "Muy bien!!";

    menuSelect(opts.length-1);
      }else if (name=='magenta') {
    document.getElementById('txtdiv').innerHTML = "intentalo de nuevo";

    menuSelect(opts.length-1);
      }else if (name=='blanco') {
    document.getElementById('txtdiv').innerHTML = "intentalo de nuevo";
```

Figura 3.28. Función javascript con el juego.

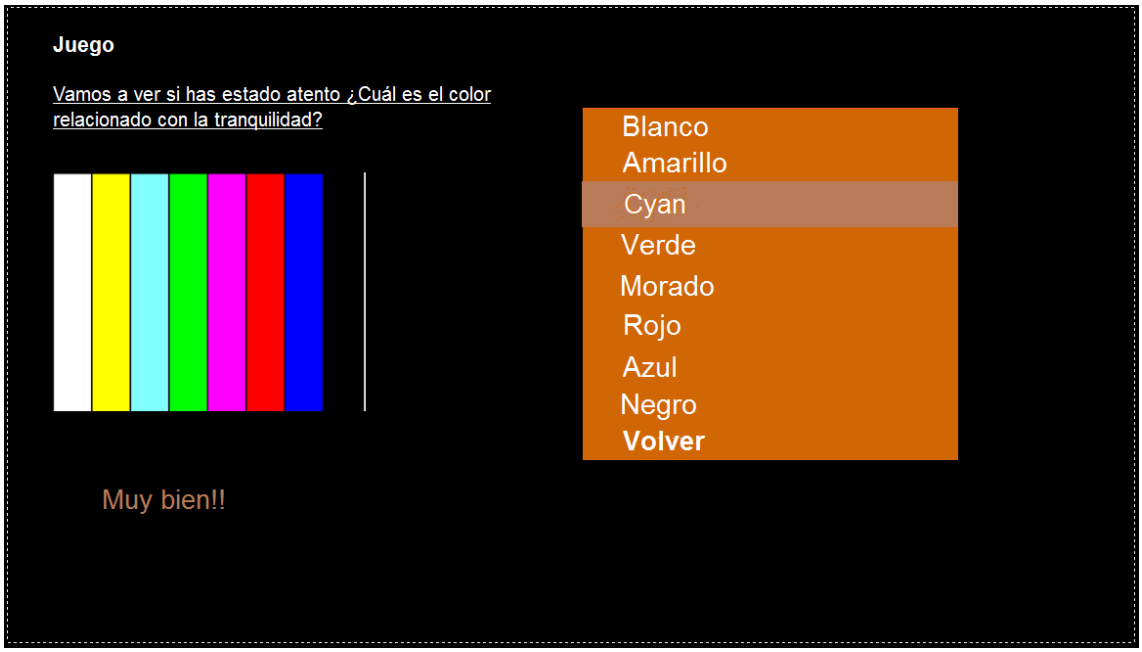


Figura 3.29. Respuesta correcta.

Figura 3.30. Respuesta incorrecta.

Por último, si el usuario lo que elige es la opción vídeo, este se encontrará con una pantalla prácticamente vacía en la que solo aparece un menú con dos opciones, la de reproducir o la de volver a la pantalla de inicio. Si el usuario elige la opción de reproducir, automáticamente comenzará a reproducirse un vídeo explicativo sobre los colores.

Para reproducir el vídeo se crea una función javascript en el programa principal que es la que se encarga de llamar el vídeo previamente codificado con los parámetros adecuados [cod].

```
function runStep(name) {
  if (name=='start') {
    try {
      var vid = document.getElementById('video');
      vid.stop();
      vid.data = 'http://itv.ard.de/video/timecode.php/video.mp4';
      vid.play(1);
      playing = true;
    } catch (e) {
```

Figura 3.31. Función javascript reproducción de vídeo.

Figura 3.32. Aspecto pantalla de reproducción de vídeo.

CAPÍTULO 4- Práctica Tecnologías e Instalaciones de Vídeo (TIV) tema 6 HbbTV

4.1. Introducción

Una vez realizadas las primeras pruebas relacionadas con la tecnología HbbTV, era el momento de comenzar a crear aplicaciones propias. Dicha tecnología ha aparecido en los últimos años y que se presupone que en un futuro aumentará su importancia en el mundo de la televisión. Por ello se pensó en que uno de los temas de la asignatura Tecnologías e Instalaciones de Vídeo fuera este, el de la televisión híbrida. De esta forma, los alumnos conocerían y estudiarían este estándar.

4.2. Objetivo

El objetivo de esta práctica era ni más ni menos que el de tener una toma de contacto con el estándar HbbTV [Spe]. Se trataba pues de que el alumno conociera este estándar, sus especificaciones, el estado del arte actual y sus posibilidades en lo relacionado con el mundo de las aplicaciones para televisión mediante tecnología híbrida.

Era necesario que el alumno adquiriera ciertas nociones básicas sobre la televisión digital conectada y en contacto sobre HbbTV para poder visualizar, modificar o crear sus propias aplicaciones.

4.3. Práctica tema 6 de TIV: "HbbTV: Interactividad en Televisión"

Un vez el alumno tenía nociones sobre la televisión híbrida y todo lo relacionado con ella, se creó una aplicación para que el alumnado pudiera ver la programación de dicha aplicación, para así poder realizar modificaciones sobre la misma.

Como en temas anteriores se había trabajado la edición de vídeo y cada uno de los grupos de alumnos había creado un anuncio, se trató de introducir este anuncio creado en la tecnología HbbTV.

La aplicación constaba de una pantalla inicial, con un menú en el que se podía seleccionar el anuncio que se deseaba visualizar, y una segunda pantalla en la que se podía reproducir el anuncio que los alumnos habían editado previamente. Dicho vídeo tenía que cumplir con las especificaciones del estándar para poder ser visualizado correctamente.

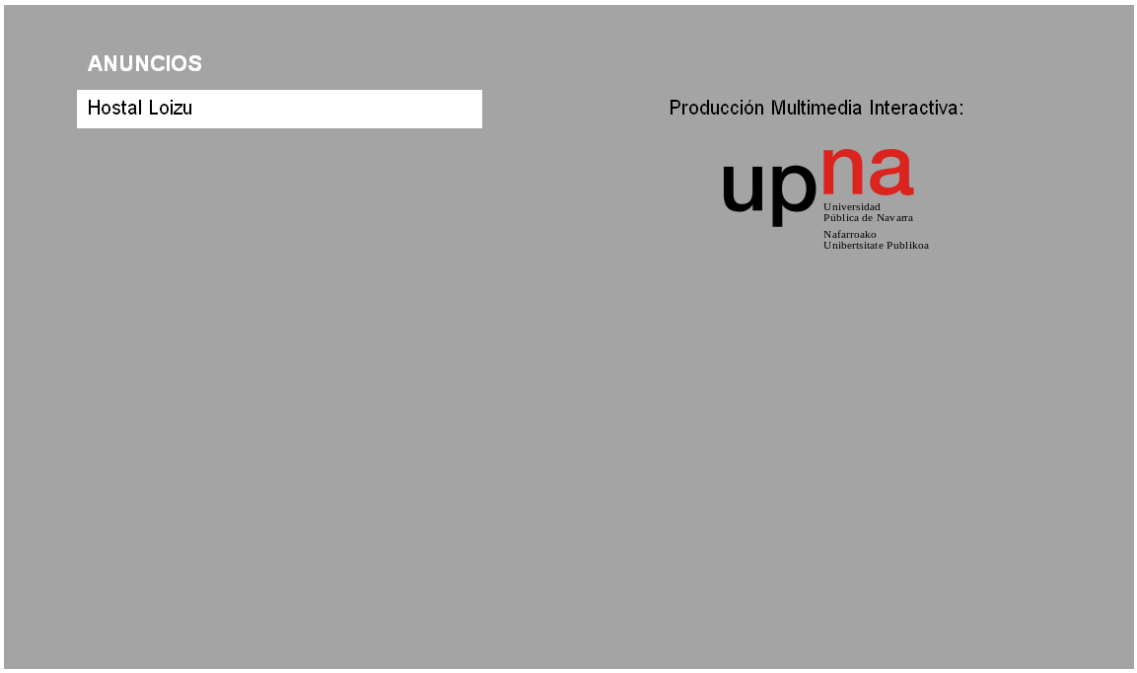


Figura 4.1. Pantalla inicial de la práctica.

Figura 4.2. Pantalla de reproducción de anuncio.

En definitiva, lo que se pedía al alumnado era realizar alguna modificación en el código planteado (como cambiar colores de fondo, tipos de letra…) y codificar su anuncio de forma correcta para poder ser visualizado.

Por todo esto, era fundamental la parte de documentación de código. Se iba a entregar a ingenieros en prácticas una documentación sobre un tema totalmente desconocido para ellos por lo que la explicación del mismo debía ser clara y concisa. Crear dicha documentación fue un trabajo exhaustivo y se

trató de explicar hasta el más mínimo detalle para que el alumno pudiera superar la práctica sin ninguna dificultad. En la siguiente figura se puede ver un ejemplo de cómo se exponía el código y su correspondiente explicación. En primer lugar aparece el código y a continuación la explicación y él porqué de ese código.

La primera de nuestras funciones es *function initVideo()*

```
function initVideo() {
  try {
    document.getElementById('video').bindToCurrentChannel();
  } catch (e) {
    // ignore
  }
  try {
    document.getElementById('video').setFullScreen(false);
  } catch (e) {
    // ignore
  }
}
```

El método **getElementById** nos retorna una referencia del objeto HTML que le pasamos como parámetro. En este caso el parámetro viene con la "id" video.

La **propiedad "id"** es un identificador único para cualquier marca HTML que luego nos permite desde Javascript acceder a dicho elemento. Es decir, luego con HTML tendremos que definir la "id" video.

A partir de este objeto accedemos a la propiedad *bindToCurrentChannel()* con la cual accedemos al canal actual.

Lo siguiente que definimos es que la propiedad *setFullScreen* sea falso, porque no queremos que el vídeo aparezca en pantalla completa.

Figura 4.3. Ejemplo de explicación de código.

La **práctica completa**, con la documentación de código y sus correspondientes explicaciones se pueden consultar en el **ANEXO III**

CAPÍTULO 5- Aplicación final

HbbTV busca una televisión interactiva que permita a los usuarios obtener contenidos bajo demanda. Por ello, en la última parte del proyecto se va a tratar de realizar una aplicación de televisión híbrida que atienda a estas necesidades del usuario de obtener servicios bajo demanda en su televisor.

5.1. Objetivo

La parte final del proyecto consiste en la creación de una aplicación HbbTV completa. Con la creación de esta aplicación lo que se busca es crear una aplicación de una situación real. Es decir, una aplicación que un usuario se pudiera encontrar en el mercado de su televisión al acceder a él con su control remoto. Como ya se ha citado anteriormente, el inicio de la televisión interactiva viene por el hecho de que las televisión bajan su índice de ventas y los programas bajan sus índices de audiencia por el hecho de que cada usuario en su ordenador puede elegir lo que quiere ver y cuando lo quiere ver cuando se le antoja. Dado que a gran cantidad de gente le gusta ver películas en su ordenador por el hecho de poder elegir el momento, el lugar y la película, esta aplicación trata de responder a esa necesidad pero sin necesidad del uso del ordenador. Consiste pues en una especie de vídeo club. El usuario entra en la aplicación y tras acceder a ella se encuentra con un menú llamado “Géneros” y es aquí donde elige el tipo de género que desea ver en ese momento. Una vez dentro del género, aparece otro menú en el que el usuario debe elegir entre una lista de películas la que desea ver en ese momento. Una vez elegida la película deseada, aparece una descripción de la misma, con actores, premios, duración de la película…Si el usuario decide verla, saltará a una página de acceso de datos. En ella deberá escribir sus datos de socio personales, y es entonces cuando el programa consulta a su base de datos. Si los datos de acceso son correctos la película comenzará a reproducirse. En caso contrario el usuario deberá introducir de nuevo sus datos de acceso. En la pantalla de reproducción el usuario tiene la opción de iniciar, parar y continuar con la reproducción para que de esta forma pueda parar en cierto momento la reproducción y no perderse nada de la película.

En el esquema siguiente se puede ver la navegación a través de la aplicación. El usuario solo puede avanzar con profundidad uno. No se puede saltar ningún paso hasta llegar a la reproducción. Sin embargo, a la hora de retroceder, el usuario puede volver hasta el punto que desee de la aplicación sin necesidad de pasar por todas las pantallas anteriormente visitadas

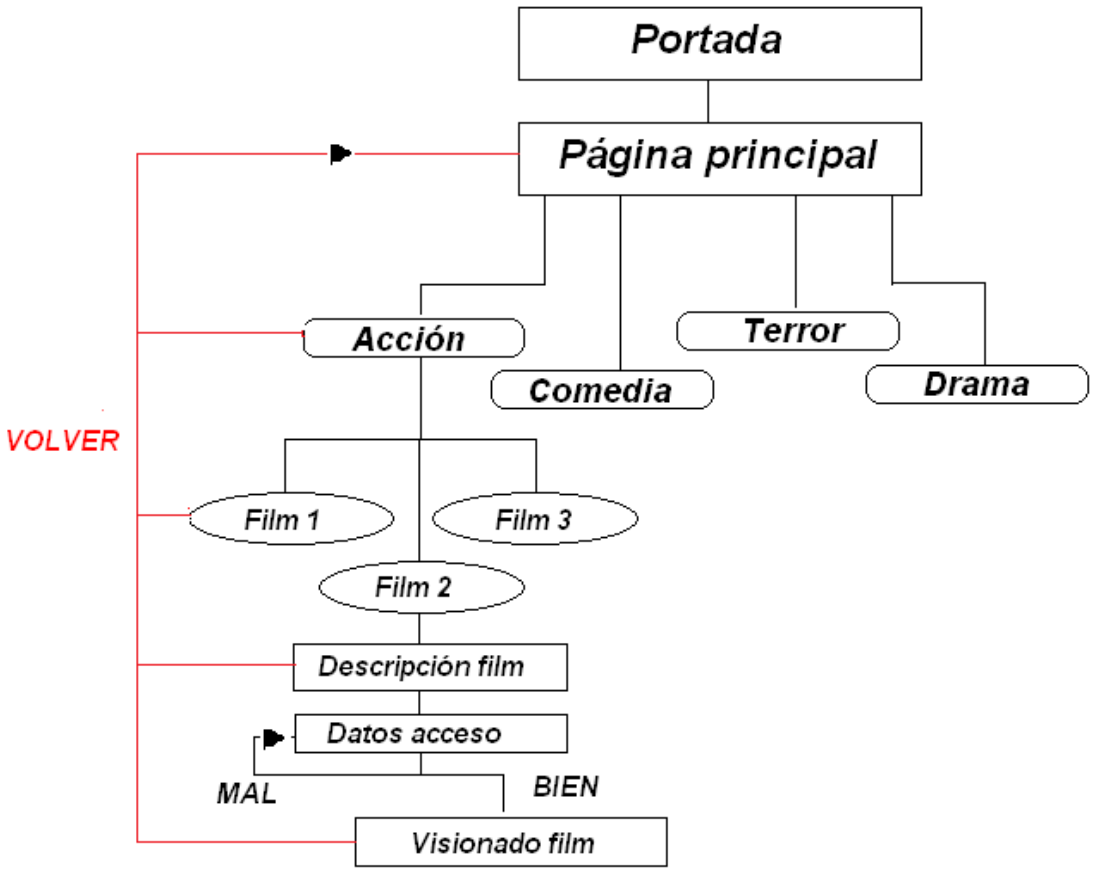


Figura 5.1. Esquema de navegación por la aplicación.

5.2. My HbbTV App

Para la creación de esta aplicación, se ha usado la misma técnica que en aplicaciones anteriores. Como se ha visto anteriormente, para la creación de una aplicación es necesario definir una base en la que se crean las hojas de estilo, las funciones globales, las librerías, la gestión de la aplicación... Por ello lo primero que se ha realizado ha sido crear dichos archivos base:

- En el archivo base de hojas de estilo se han definido los diferentes tipos de menú, de formatos de letra, de cuadros de imagen…En concreto, para esta aplicación han sido necesarios 5 diferentes tipos de menú, así como 6 diferentes formatos de letra. En cuanto a los cuadros de imagen es necesario definirlos tan solo una vez puesto que el tamaño de los mismos se puede definir desde el código HTML del programa principal.

- En el archivo de base javascript se han definido las diferentes funciones globales que posteriormente son utilizadas en la aplicación. En este documento aparecen funciones como la de iniciar la aplicación

- En el archivos base de php se definen los contenidos de la aplicación, las versiones de los estándares , la gestión de la aplicación y un objeto de vídeo que posteriormente es el que permita la reproducción de la película.

- Se ha utilizado la librería anteriormente citada de equivalencia entre los botones del teclado del ordenador y los botones del control remoto.

Una vez definida esta base, comienzan a construirse los diferentes archivos para cada una de las pantallas de las que va a disponer la aplicación. Estos archivos llevan el nombre de index.php y se crea uno de ellos para cada una de las páginas de las que disponga la aplicación.

5.2.1. Pantalla de inicio

La pantalla de inicio es muy simple y sirve a modo de presentación. En ella aparecen una imagen con la marca del vídeo club, un botón de acceso a la aplicación y en la parte inferior dos botones. Uno verde para volver al inicio y otro rojo para salir de la aplicación, de forma que el usuario con su control remoto pueda volver al inicio o salir de la aplicación pulsando los botones verde o rojo.

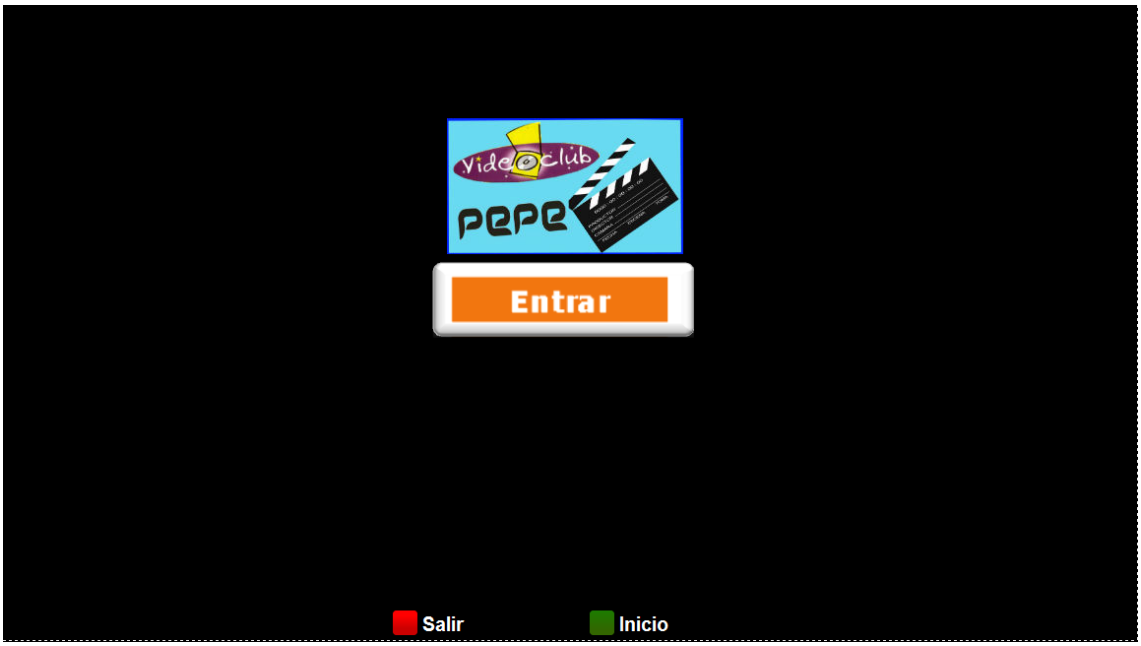


Figura 5.2. Aspecto de la pantalla inicial.

En cuanto a la programación de esta página, en primer lugar se ha introducido una imagen con el logo del vídeo club. Esto se ha realizado en la parte de HTML del programa y en él se define tanto la imagen como la posición de la misma en la página.

```
<div style="left: 455px; top: 100px; width: 450px; height: 300px; background-color: #000000;">
  <div class="imgdiv" style="left: 60px; top: 54px; width:240px; height: 200px; background-image: url(definitiva.jpg);"></div>
</div>
```

Figura 5.3. Código HTML imagen de marca de vídeo club.

En cuanto al botón de acceso a la siguiente pantalla, la forma de la misma ha sido definida en el archivo base de hojas de estilo, y posteriormente ha sido definido en la parte de código HTML del programa.

```
ul.menu2 {
  padding: 0;
  margin: 0;
  position: absolute;
  width: 100px;
}
.menu2 li {
  display: inline;
  overflow: hidden;
  float:left;
  width: 255px;
  height: 72px;
  text-align: left;
  padding-left: 11px;
  padding-top: 5px;
  font: 20px sans-serif;
  color: #000000;
  background-image: url(BotonEntrar.png);
}
```

Figura 5.4. Definición del botón en la hoja de estilos.

```
<ul id="menu" class="menu2" style="left: 500px; top: 300px;">
  <li name="GENEROS" ></li>
</ul>
```

Figura 5.5.Código HTML colocación del botón de acceso.

Por último, para la colocación de los botones de vuelta a inicio y de salida de aplicación, se ha definido una función variable en javascript, que es la que se encarga de mandar a la dirección adecuada según la selección del usuario. Se han colocado imágenes de color rojo y verde para que el usuario sepa cuál es el botón que debe presionar. Esta parte se ha hecho nuevamente en la parte de código HTML del programa.

```
} else if (kc==VK_RED){
  closeApp();
  return true;

} else if (kc==VK_GREEN){
document.location.href="http://130.206.170.120/TIV/ImanolEslava/VIDEO CLUB APP/programacion";
return true;
```

Figura 5.6. Función javascript de botones verde y rojo.

Como se ve en la imagen, al pulsar el botón rojo, el programa llama a otra función llamada close app. En este caso esta función ha sido creada para que el usuario no pueda salir de la aplicación y si el mismo pulsara el botón rojo lo que hace esta función es introducir un mensaje de alerta que dice “cannot exit application”.

```
function closeApp() {
  try {
    var app = document.getElementById('appmgr').getOwnerApplication(document);
    app.destroyApplication();
    return;
  } catch (e) {
    alert('Cannot exit application');
  }
}
```

Figura 5.7. Función closeApp.

```
<div style="left: 400px; top: 600px; width: 200px; height: 100px; background-color: #000000;">
<div class="txtdiv txtlg" style="left: 90px; top: 55px; width: 100px; height: 30px;">Salir</div>
  <div class="imgdiv" style="left: 60px; top: 54px; width:25px; height: 25px; background-image: url(btn-rojo.png);"></div>
</div>
<div style="left: 600px; top: 600px; width: 200px; height: 100px; background-color: #000000;">
<div class="txtdiv txtlg" style="left: 90px; top: 55px; width: 300px; height: 30px;">Inicio</div>
  <div class="imgdiv" style="left: 60px; top: 54px; width:25px; height: 25px; background-image: url(btn-verde.png);"></div>
</div>
```

5.8. Programación HTML de botones verde y rojo.

5.2.2. Pantalla géneros

En esta pantalla es donde realmente comienza la aplicación. En ella aparecen diferentes componentes. Encontramos un menú de imágenes, un cuadro de texto con indicaciones, otro cuadro de texto con los datos de contacto del vídeo club, una imagen de la marca, los botones verde y rojo antes nombrados y por último un cuadro de texto variable pues el texto que aparece en el varía en función de la selección en la que se encuentra el usuario.

Figura 5.9. Pantalla de géneros.

El menú consta de cinco imágenes, que son cada uno de los géneros que oferta la aplicación. Este menú ha sido creado en la hoja de estilos y definido en la parte de código HTML del programa. Se ha buscado que una sección roja marque en qué posición del menú se encuentra el usuario.

```
ul.menu {
    padding: 0;
    margin: 0;
    position: absolute;
    width: 100px;
}
.menu li {
    display: inline;
    overflow: hidden;
    float:left;
    width: 115px;
    height: 190px;
    text-align: left;
    padding-left: 11px;
    padding-top: 5px;
    font: 20px sans-serif;
    color: #000000;
    background-color: #000000;
}
.menu .lisel {
    color: #000000;
    background-color: #ff4000;
}
```

Figura 5.10. Menú de imágenes en géneros.

Para este menú se elige un tamaño concreto de cada uno de los componentes del mismo, y esto va en función del tamaño que se le quiera dar a cada una de las imágenes que componen dicho menú.

```
<ul id="menu" class="menu" style="left: 111px; top: 100px; width: 900px; height: 150px;">
  <li name="COMEDIA" COMEDIA<br></br><img  src="comedia.PNG" style=" position: absolute;" /> </li>
  <li name="TERROR"  TERROR<br></br><img  src="terror.PNG" style=" position: absolute;" /> </li>
  <li name="ACCION"  ACCIÓN  <br></br><img  src="accion.PNG" style=" position: absolute;" /> </li>
  <li name="AMOR"    AMOR <br></br><img  src="amor.png" style=" position: absolute;" /> </li>
  <li name="CIENCIAFICCION" C.FICCIÓN  <br></br> <img src="cienciaficcion.png" style=" position: absolute;" />
</ul>
```

Figura 5.11. Definición menú en código HTML.

Para la navegación a través del menú mediante teclado, se han añadido sentencias a la función de botón rojo y verde que antes se había creado. Estas sentencias permiten al usuario cambiar la selección del menú con las flechas del control remoto.

```
function handleKeyCode(kc) {
  if (kc==VK_UP) {
    menuSelect(selected-3);
    setDescr();
    return true;
  } else if (kc==VK_DOWN) {
    menuSelect(selected+3);
    setDescr();
    return true;
  } else if (kc==VK_LEFT){
    menuSelect(selected-1);
    setDescr();
    return true;
  } else if (kc==VK_RIGHT){
    menuSelect(selected+1);
    setDescr();
```

Figura 5.12. Funcion javascript de navegación mediante flechas.

El resto de componentes han sido creados en la parte de HTML de código del programa principal, tanto los cuadros de texto como la imagen. Para los botones verde y rojo el proceso es el mismo al seguido en la página anterior.

```
<div style="left: 750px; top: 100px; width: 450px; height: 500px; background-color: #000000;">
  <div class="txtdiv txtlh" style="left: 60px; top: 200px; width: 400px; height: 300px; color: #ffffff;">
  Hazte socio ya <br></br><br></br>en nuestro local :<br></br><br></br>TLF:600000006<br></br><br></br>C.P. 00000<br></br><br></br>Pamplona</di
  <div class="imgdiv" style="left: 60px; top: 4px; width:240px; height: 200px; background-image: url(definitiva.jpg);"></div>
</div>
```

Figura 5.13. Definición de cuadro de texto e imagen.

Por último, destacar que se crea un cuadro de texto que va variando en función de la posición del menú sobre la que se encuentra el usuario. Dentro del cuadro de texto, se crea una variable con una id, y más tarde en cada una de

las partes del menú se iguala esa variable con el texto que se desea para cada opción.

```
<div class="txtdiv" style="left: 111px; top: 300px; width: 450px; height: 500px;"><u>Elige tu género</u><br />
Selecciona el tipo de género que más se adecue a tus apetencias<br /><br />
<u>Descripción</u><br />
<span id="descr"> </span>
</div>
```

variable

Figura 5.14. Definición ID descr.

```
<ul id="menu" class="menu" style="left: 111px; top: 100px; width: 900px; height: 150px;">
  <li name="COMEDIA"    descr="Comedia:Son alegres y deliberadamente diseñadas para divertir
  <li name="TERROR"     descr="Terror: Se caracteriza por su voluntad de provocar en el espe
  <li name="ACCION"   descr="Acción: Por lo general incluyen altas dosis de adrenalina y ene
  <li name="AMOR"      descr="Amor: se caracteriza por retratar argumentos construidos de eve
  <li name="CIENCIAFICCION"  descr="Ciencia Ficción: Utiliza representaciones especulativas
</ul>
```

ID descr

Figura 5.15. Igualdad de descr con cada descripción.

De esta forma, la descripción del género va variando en función del género que tiene seleccionado el usuario de su menú. En caso de no saber qué tipo de película desea visualizar esta breve descripción puede ayudar al usuario a decidir qué tipo de género es el que desea ver en ese momento.

5.2.3. Pantalla género concreto: Acción

En esta nueva pantalla, aparece un nuevo menú de imágenes, una breve descripción de la película, que como anteriormente varía según la opción del menú seleccionada, y nuevamente tanto el botón verde y el rojo, además de un nuevo botón amarillo que permite al usuario mediante ese botón volver a la pantalla de géneros.

Figura 5.16. Pantalla de género acción.

Como se ve en la imagen, el tamaño de las imágenes es mayor que en la pantalla anterior, por lo que es necesario crear un nuevo menú en el archivo base de hojas de estilo que atienda a las necesidades de este caso.

```
ul.menu3 {
    padding: 0;
    margin: 0;
    position: absolute;
    width: 100px;
}
.menu3 li {
    display: inline;
    overflow: hidden;
    float:left;
    width:222px;
    height: 282px;
    text-align: left;
    padding-left: 11px;
    padding-top: 5px;
    font: 20px sans-serif;
    color: #000000;
    background-color: #000000;
}
.menu3 .lisel {
    color: #ffffff;
    background-color: #ff0000;
}
```

Figura 5.17. Código css para menú tipo 3.

En cuanto al botón amarillo que permite volver a la pantalla de géneros, ha sido necesario definir unas nuevas sentencias en la función javascript anteriormente creada para el resto de botones, y a su vez colocar un cuadro de imagen con el botón amarillo para que el usuario sepa cuál es el botón que debe pulsar si desea volver a la pantalla de géneros.

```
  } else if (kc==VK_YELLOW){
document.location.href="http://130.206.170.120/TIV/ImanolEslava/VIDEO%20CLUB%20APP/programacion/GENEROS";
return true;
```

Figura 5.18. Código javascript con función del botón amarillo.

```
<div style="left: 700px; top: 600px; width: 220px; height: 100px; background-color: #000000;">
<div class="txtdiv txtlg" style="left: 90px; top: 55px; width: 300px; height: 40px;">Géneros</div>
  <div class="imgdiv" style="left: 60px; top: 54px; width:25px; height: 25px; background-image: url(btn-amarillo.png);"></div>
</div>
```

Figura 5.19. Código de cuadro de imagen con botón amarillo.

Por último para la descripción de cada una de las películas se ha utilizado el mismo formato que en la pantalla anterior, creando una ID con una variable y cambiando esa variable en cada una de las opciones del menú.

Figura 5.20. Pantalla género acción con distinta selección.

Si se compara la imagen anterior con la imagen 5.16, se observa como el texto de la descripción del argumento ha variado. A continuación aparecen una serie de imágenes que representan la pantalla de cada uno de los cinco géneros creados.

Figura 5.21. Pantalla género comedia.

Figura 5.22. Pantalla género amor.

Figura 5.23. Pantalla género ciencia ficción.

Figura 5.24. Pantalla género terror.

5.2.4. Pantalla de película

En este caso, la pantalla que se encuentra el usuario consta de una imagen principal de la película escogida, de una descripción de la misma con actores, premios, argumento…También se encuentra un botón para iniciar la reproducción y finalmente los botones rojo, verde y amarillo antes definidos además de un nuevo botón, en este caso de color azul, que permite volver al género en el que se encontraba el usuario.

Figura 5.25. Pantalla de película Acción Civil.

Para la colocación tanto de la imagen principal de la película como de la descripción de la misma, se crean cuadros de imágenes y se colocan en la posición deseada.

```
<div style="left: 540px; top: 60px; width: 587px; height: 567px; background-color: #000000;">
  <div class="imgdiv" style="left: 10px; top: 1px; width:651px; height: 628px; background-image: url(descripcion.PNG);"></div>
</div>
<div style="left: 111px; top: 100px; width: 400px; height: 401px; background-color: #000000;">
  <div class="imgdiv" style="left: 1px; top: 1px; width:398px; height: 400px; background-image: url(travolta.jpg);"></div>
</div>
```

Figura 5.26. Código HTML de cuadros de imagen.

Como se ha realizado en el caso anterior, para añadir el botón azul de vuelta al género, se han añadido ciertas sentencias a la función javascript anteriormente creada y se ha definido un nuevo cuadro de imagen del botón azul para que el usuario sepa cuál es el botón que desea pulsar si quiere volver al género en el que se encontraba.

```
  } else if (kc==VK_BLUE){
document.location.href="http://130.206.170.120/TIV/ImanolEslava/VIDEO%20CLUB%20APP/programacion/GENEROS/ACCION";
return true;
```

Figura 5.27. Código javascript de botón azul.

```
<div style="left: 800px; top: 600px; width: 230px; height: 100px; background-color: #000000;">
<div class="txtdiv txtlg" style="left: 90px; top: 55px; width: 300px; height: 30px;">Acción</div>
  <div class="imgdiv" style="left: 60px; top: 54px; width:25px; height: 25px; background-image: url(btn-azul.png);"></div>
</div>
```

Figura 5.28. Código HTML para visualización de botón azul.

Como es lógico la función de este botón azul es diferente para cada uno de los géneros, pues cada película tiene que volver a su género adecuado, por lo que en función del género al que se desee enlazar la película hay que hacer variaciones tanto en la parte de javascript como en la parte de HTML. A continuación se muestra una imagen de otra película de diferente género y si se compara con la imagen 5.25. se comprueba que el botón azul ha variado su función. En la imagen 5.25. se da la opción de volver al género acción y en la imagen siguiente la opción que da es la de volver al género comedia.

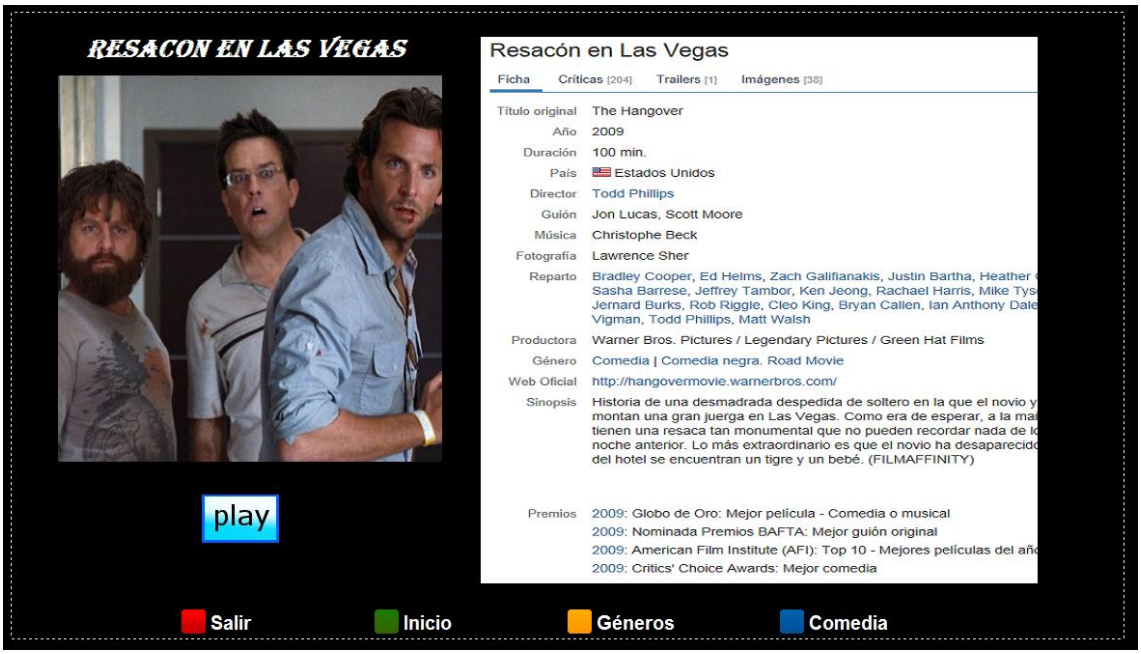


Figura 5.29. Película de género comedia.

Por último, en esta pantalla el usuario se encuentra con un botón que se llama PLAY. Este botón nuevo por lo que es creado en la hoja de estilos y posteriormente definido en el código HTML del programa.

```
ul.menu4 {
  padding: 0;
  margin: 0;
  position: absolute;
  width: 100px;
}
.menu4 li {
  display: inline;
  overflow: hidden;
  float:left;
  width: 154px;
  height: 73px;
  text-align: left;
  padding-left: 11px;
  padding-top: 5px;
  font: 20px sans-serif;
  color: #000000;
  background-image: url(Play.png);
}
```

Figura 5.30. Creación del botón PLAY en hoja de estilos.

```
<ul id="menu" class="menu4" style="left: 250px; top: 530px;">
   <li name="GENEROS" ><img src="play.jpg" style=" position: absolute;" /></li>
</ul>
```

Figura 5.31. Definición del botón en código HTML.

Es de esperar que al pulsar dicho botón, el programa pase a una nueva pantalla en la que comience la reproducción. Pero previamente el usuario debe identificarse como socio del vídeo club. Para que suceda esto, se ha creado una función javascript llamada *ventanaSocio* que salta al ser pulsado el botón de play. Lo que hace esta función es llamar a un formulario de identificación.

```
} else if (kc==VK_ENTER) {
  ventanaSocio();
  return true;
```

Figura 5.32. Llamada a función ventanaSocio al pulsar botón PLAY.

```
function ventanaSocio() {
  try {
    var app = document.getElementById('appmgr').getOwnerApplication(document);
    app.destroyApplication();
    return;
  } catch (e) {
    document.location.href="http://130.206.170.120/TIV/ImanolEslava/VIDEO%20CLUB%20APP
    /programacion/GENEROS/ACCION/ACCION CIVIL/formulario.php";
  }
```

Figura 5.33. Función javascript ventanaSocio.

5.2.5. Identificación

Una vez el usuario elige la película que desea ver y pulsa el botón de reproducir, es necesario que este introduzca sus datos para poder ver el film. Se encuentra con un formulario, que si está bien completado le permite reproducir la película pero que si no se rellena con los datos correctos, el programa le devuelve a la pantalla de la película.

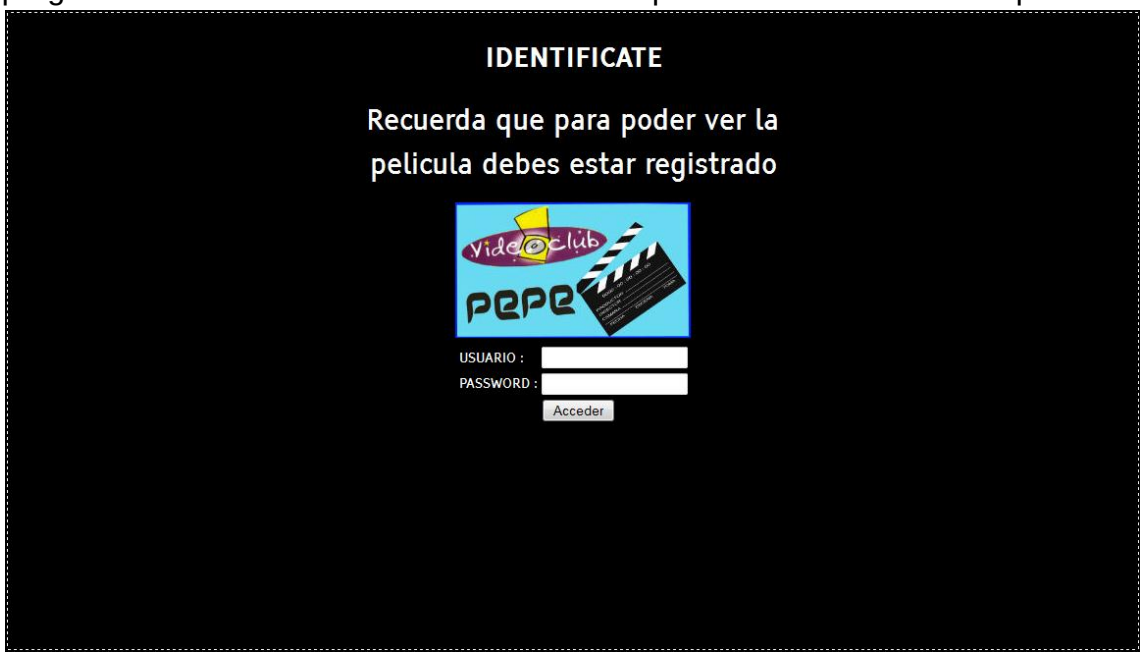


Figura 5.34 Pantalla de identificación.

Para crear este formulario se ha utilizado programación en HTML, creando dos campos a rellenar (el de usuario y contraseña) y un botón que al ser pulsado envía la petición.

```
<p><font color =white size=6>Recuerda que para poder ver la <br>pelicula debes estar registrado</font></p>

<IMG SRC="definitiva.jpg"/>


    <form action="login.php" method="post" >
        <center><table>
        <tr>
            <td><font color =white >USUARIO :</font></td>
            <td><input type="text" class="input_field" name="username"/></td>
        </tr>

        <tr>
            <td><font color =white >PASSWORD :</font></td>
            <td><input class="input_field" type="password" name="password"/></td>
        </tr>
        <tr>
            <td></td>
            <td><input type="submit" value="Acceder"/></td>
        </tr>
```

Figura 5.35. Formulario HTML.

Como se ve en la imagen, el formulario consta simplemente de 3 entradas, una de ellas de tipo campo de entrada, otra de ellas de tipo contraseña y la última de ellas de tipo enviar. Lo más importante de este formulario es que tiene una llamada a otro archivo como se puede ver en las primeras líneas de la imagen. Desde aquí se llama a otro archivo de nombre *login.php* que será el que decida si el usuario ha introducido de forma correcta sus datos.

Dicho archivo login.php es un archivo de consulta a base de datos. Se ha creado mediante la aplicación PHPMyAdmin [MyA] una base de datos con los datos de los clientes del vídeo club. Este archivo lo que hace es tomar los datos introducidos por el usuario y compararlos con los datos de la base de datos. Si coinciden significa que el usuario está registrado y por tanto puede ver la película. Si no coinciden el usuario no está registrado y por tanto no puede acceder a la reproducción.

A continuación se muestra la imagen de la base de datos creada en PHPMyAdmin para el vídeo club.

	id	username	password
	1	usuario	usuario
	2	imanol	eslava
	3	pedro	pedro
	4	juan	juan

Figura 5.36. Base de datos de vídeo club.

```php
function Conectarse()
{
if (!($link=mysql_connect("localhost","Imanol","imanol")))
    {
    echo "error conectando a la base de datos.";
    exit();
    }
    if (!mysql_select_db("prueba",$link))
    {
    echo "error seleccionando la base de datos";
    exit();
    }
    return $link;
}
```

Figura 5.37. Función para conectarse a base de datos concreta.

Con esta función lo que se hace es conectarse a la base de datos del vídeo club. En caso de que no se pueda realizar la conexión aparecerá un mensaje de error.

```php
$username=$_POST['username'];
$password=$_POST['password'];
$sesion_login= true;
$con=Conectarse();
$query=" SELECT * FROM pruebausuario WHERE username = '".$username."' AND password ='".$password."'";
$q=mysql_query($query,$con);
```

Figura 5.38. Código de selección de datos de la base de datos.

Esta parte del código se encarga de buscar en la base de datos los campos de usuario y contraseña para así poder compararlos con los introducidos.

```php
try{

if(mysql_result($q,0))
{$result = mysql_result($q,0);

    header('Location:http://130.206.170.120/TIV/ImanolEslava/VIDEO%20CLUB%20APP
        /programacion/GENEROS/ACCION/ACCION%20CIVIL/REPRODUCIR/index.php')

  } else
    header('Location:http://130.206.170.120/TIV/ImanolEslava/VIDEO%20CLUB%20APP
        /programacion/GENEROS/ACCION/ACCION%20CIVIL/index.php');
  }catch(Exception $error){}
mysql_close($con);
```

Figura 5.39. Condición y comparación entre datos de entrada y base de datos.

Esta parte de código es la que decide si el usuario esta registrado o no. Si se cumple la condición el usuario es redirigido a la pantalla de reproducción y i no

es así es redirigido a la pantalla anterior. En caso de que no pueda realizar la comparación aparecerá un mensaje de error.

Para el uso de tanto MySQL, Apache y PHP se utiliza un servidor XAMPP [Xam]. XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl.

El programa está liberado bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas.

5.2.6. Reproducción

En esta página, lo que nos encontramos en un menú con tres imágenes. Y los cuatro botones rojo, amarillo, verde y azul antes definidos.

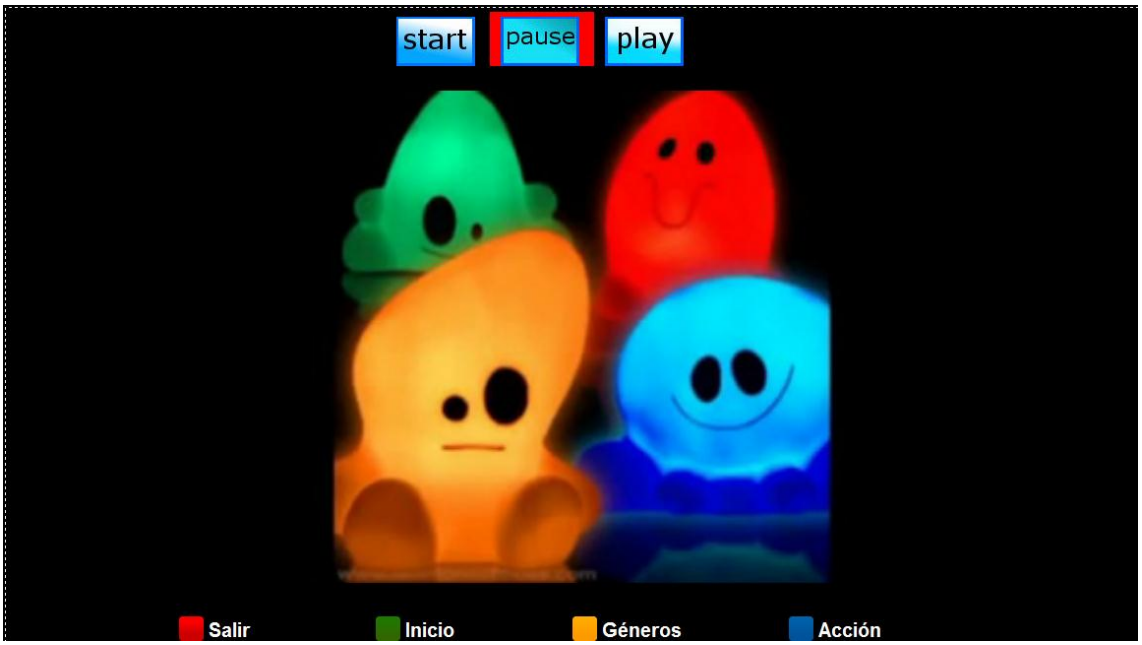


Figura 5.40. Pantalla de reproducción.

Este nuevo menú horizontal ha sido creado en la hoja de estilos base y cuenta con tres botones que permiten comenzar la reproducción, pararla y reanudarla.

```
ul.menu5{
  padding: 0;
  margin: 0;
  position: absolute;
  width: 100px;
}
.menu5 li {
  display: inline;
  overflow: hidden;
  float:left;
  width:95px;
  height: 50px;
  text-align: left;
  padding-left: 11px;
  padding-top: 5px;
  font: 20px sans-serif;
  color: #000000;
  background-color: #000000;
}
.menu5 .lisel {
  color: #ffffff;
  background-color: #ff0000;
}
```

Figura 5.41. Creación de menú para reproducción.

```
<ul id="menu" class="menu5" style="left: 450px; top: 40px; width: 900px; height: 50px;">
  <li name="start"><img src="start.jpg" style=" position: absolute;" /></li>
  <li name="pause"><img src="pause.jpg" style=" position: absolute;" /></li>
  <li name="play"> <img src="play.jpg"  style=" position: absolute;" /></li>
</ul>
```

Figura 5.42. Definición de menú en HTML.

Lo más importante, y más diferente de esta página es la aparición del vídeo. La función que controla el vídeo es *function runStep(name).*

En esta función lo primero que aparece es una condición, que es que si no se pulsa el ENTER sobre la opción de empezar a ver el vídeo no entra en la condición.

A continuación se define una variable vid a la que se iguala con la id de vídeo, y se le definen los atributos de stop, data y play. La propiedad **data** será la que se iguale con la **dirección en la que está la película**. Más adelante se explicará el formato en el que tendrá que estar el vídeo. La propiedad play es la que marca la velocidad de reproducción.

Hay otras dos opciones en el condicional, que son la de pause y play, llaman a una función setSpeed. Esta función es creada para poder poner velocidad a la reproducción del vídeo. En este caso ponemos valor 0 para pause y 1 para volver a reproducir pero se podría poner velocidad doble o mitad.

Por último se pone la variable playing a true, pues antes la había sido definida como falsa.

```
function runStep(name) {
  if (name=='start') {
    try {
      var vid = document.getElementById('video');
      vid.stop();
      vid.data = "http://130.206.170.120/TIV/ImanolEslava/VIDEO%20CLUB%20APP
    /programacion/GENEROS/ACCION/ACCION%20CIVIL/REPRODUCIR/colores.mp4";
      vid.play(1);
      playing = true;
    } catch (e) {

    }
  } else if (name=='pause') {
    setSpeed(0);
  } else if (name=='play') {
    setSpeed(1);
  }
}
```

Figura 5.43. Función javascript runStep.

```
function setSpeed(fact) {

    var vid = document.getElementById('video');
    vid.play(fact);
}
```

Figura 5.44. Función javascript setSpeed.

En la parte de código HTML del programa es necesario definir un contenedor de vídeo con la longitud de la pantalla de vídeo y con la situación de la misma en la pantalla.

```
<object id="video" type="video/mp4" style="position: absolute;
left: 200px; top: 120px; width: 900px; height: 500px;"></object>
```

Figura 5.45. Definición de contendor de vídeo.

A continuación se presentan los formatos de vídeo soportados por HbbTV.

Dichos formatos de vídeo soportados por el estándar HbbTV están especificados por la especificación OIPF [OP3]. Se pueden encontrar en la sección 5 de este documento. Vemos una tabla de formatos de vídeo y audio para 25 Hz.

System Format	Video Format	Audio Format	Mime Type
TS	AVC_HD_25 AVC_SD_25 AVC_SP_25	HEAAC HEAAC2 HEAAC_MPS MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3 DTS	video/mpeg or video/mp2t
TTS	AVC_HD_25 AVC_SD_25 AVC_SP_25	HEAAC HEAAC2 HEAAC_MPS MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3 DTS	video/vnd.dlna.mpeg-tts
MP4	AVC_HD_25 AVC_SD_25 AVC_SP_25	HEAAC HEAAC2 HEAAC_MPS MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3 DTS	video/mp4
TS	MPEG2_SD_25 MPEG2_SP_25	MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3	video/mpeg or video/mp2t
TTS	MPEG2_SD_25 MPEG2_SP_25	MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3	video/vnd.dlna.mpeg-tts

Table 1: A/V Media Formats for 25Hz video system

Aquí la misma tabla para formato de 30 Hz:

System Format	Video Format	Audio Format	Mime Type
TS	AVC_HD_30 AVC_SD_30 AVC_SP_30	HEAAC HEAAC2 HEAAC_MPS AC3 E-AC3 DTS	video/mpeg or video/mp2t
TTS	AVC_HD_30 AVC_SD_30 AVC_SP_30	HEAAC HEAAC2 HEAAC_MPS AC3 E-AC3 DTS	video/vnd.dlna.mpeg-tts
MP4	AVC_HD_30 AVC_SD_30 AVC_SP_30	HEAAC HEAAC2 HEAAC_MPS AC3 E-AC3 DTS	video/mp4

Table 2: A/V Media Formats for 30Hz video system

El formato MP4 está definido en la Parte 14 de MPEG-4 [MPG]. Es un formato estándar de contenedor multimedia especificado como parte del estándar IEC. Se utiliza para almacenar los formatos audiovisuales especificados por ISO/IEC y el grupo MPEG (Moving Picture Experts Group) al igual que otros formatos audiovisuales. Se utiliza típicamente para almacenar datos en archivos para ordenadores, para transmitir flujos audiovisuales y en muchas otras aplicaciones.

Comúnmente se utiliza para encapsular contenido de audio y vídeo digital, pero también puede ser utilizado para combinar muchos más tipos de contenido multimedia, tales como audio múltiple, vídeos, subtítulos e imágenes fijas.

En nuestro caso hemos utilizado un contenedor MP4 que contenía vídeo codificado con H.264 (o lo que es lo mismo, MPEG-4 Parte 10 [cod]), que sí está soportado por HbbTV. H.264 es una norma que define un códec de vídeo de alta compresión. La intención del proyecto H.264/AVC fue la de crear un estándar capaz de proporcionar una buena calidad de imagen con tasas binarias notablemente inferiores a los estándares previos (MPEG-2, H.263 o MPEG-4 parte 2), además de no incrementar la complejidad de su diseño.

Conclusiones

En primer lugar, destacar que la realización de este proyecto ha sido enriquecedora a nivel tanto intelectual como personal. El conseguir el objetivo inicial ha sido muy gratificante y la realización de este trabajo me será de gran utilidad en el futuro.

En este proyecto he tenido la oportunidad de ponerme en contacto con gran cantidad de tecnologías que son las utilizadas en la actualidad para la creación tanto de entornos webs, como de aplicaciones móviles:

- **Html + Css:** html es un lenguaje que define el contenido de una página web, como texto, imágenes, etc. Css hace referencia a un lenguaje de hojas de estilo usado para describir el aspecto y formato de un documento escrito en html.
- **Javascript:** lenguaje de programación interpretado. Se utiliza principalmente en su forma del lado del cliente. Es dinámico y se suele implementar como parte de un navegador permitiendo mejores en la interfaz de usuario y páginas web dinámicas.
- **Php:** es un lenguaje de programación de código del lado del servidor, creado para el desarrollo web de contenido dinámico.
- **Sql:** es un lenguaje de declarativo de acceso a bases de datos y que permite especificar diversos tipos de operaciones sobre ellas.
- **MySql:** es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. Se desarrolla como software libre en un esquema de licenciamiento dual.
- **PhpMyAdmin:** es una herramienta escrita en php con la intención de manejar la administración de MySQL a través de páginas web utilizando internet. Actualmente puede crear y eliminar bases de datos, crear y eliminar tablas y campos así como ejecutar cualquier sentencia SQL.

Durante el transcurso de la carrera no había tenido la oportunidad de estudiar estas tecnologías por lo que este proyecto me ha permitido ampliar mis conocimientos de cara a mi salida a la vida laboral.

El hecho de tener que dedicar gran parte del tiempo a estas tecnologías ha frenado a la vez el mayor desarrollo de este proyecto, ya que como es lógico no se podían crear grandes aplicaciones sin saber prácticamente nada de programación.

Pese a todo, la creación de la aplicación final ha sido satisfactoria. Se ha implementado una aplicación de entidad, utilizando todas las tecnologías estudiadas y con una navegación correcta. Se ha conseguido que en la misma aparezca un entorno real de una aplicación de este tipo, que se hagan

consultas a bases de datos de forma satisfactoria y que los vídeos sean reproducidos de forma correcta.

Hubiera sido positivo poder mandar contenido en streaming, o también introducir alguna forma de interacción directa del usuario como la de un widget de twitter por ejemplo.

Otro de los objetivos que quedan pendientes es el de poder lanzar la aplicación a una televisión real. Esto no ha sido posible porque los medios disponibles en la universidad no lo han permitido.

Para terminar, una pequeña valoración personal de la tecnología HbbTV. En mi opinión, HbbTV es una oportunidad de crear un estándar global, pues es abierto, utiliza tecnologías ya existentes para implementar sus aplicaciones y permite al usuario disfrutar de una televisión más completa con un nuevo mundo de navegación e interacción con su televisor. El problema de esta tecnología es que es abierta, lo que supone un problema para las empresas, que disponen de su propia tecnología y se cierran a trabajar de forma conjunta. Dichas empresas prefieren trabajar con su tecnología y de esta forma ofertar aplicaciones que solo pueden ser visualizadas por sus dispositivos. En esta lucha, se definirá si HbbTV triunfa o no.

BIBLIOGRAFÍA

[Est] – Estudio realizado por Iñaki Úcar
[ETS] – ETSI TS 102 796 v1.1.1 "*Hybrid Broadcast Broadband TV*"
[Spe] – *HbbTV Specification Version 1.5*
[aed] – AEDETI: http://www.aedeti.com/
[MPE] – MPEG DASH Wikipedia:
http://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP
[Hbb] – Consorcio HbbTV: http://www.hbbtv.org
[pro] – Protocolos TCP/IP:
http://es.wikipedia.org/wiki/familia_de_protocolos_de_internet
[pro2] – Protocolo HTTP:
http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[jav] - JavaScript: http://es.wikipedia.org/wiki/JavaScript
[htm] - HTML: http://es.wikipedia.org/wiki/HTML
[php] - PHP: http://es.wikipedia.org/wiki/PHP
[css] - CSS: http://es.wikipedia.org/wiki/CSS
[sql] – SQL: http://es.wikipedia.org/wiki/sql
[MHP] – MHP: http://es.wikipedia.org/wiki/MHP
[Stv] – Smart TV: http://es.wikipedia.org/wiki/Smarttv
[OIP] - OIPF: http://www.oipf.tv/
[MPG] - MPEG-4 Parte 14 o M4A: http://es.wikipedia.org/wiki/MPEG-4_Parte_14
[Cod] - Codificación H.264: http://es.wikipedia.org/wiki/MPEG_4_Parte_10
[Ora] - Oracle Virtual box: https://www.virtualbox.org/
[CE] – CE-HTML Wikipedia: http://en.wikipedia.org/wiki/CE-HTML
[DVB] – Digital Video Broadcasting: http://es.wikipedia.org/wiki/DVB
[W3c] – World Wide Web consortium: http://es.wikipedia.org/wiki/W3c
[Op1] – Open IPTV Forum – *Release 1 DAE Reference Guide* v1.0
[Op2] – Open IPTV Forum – *Release 1 Specification volume 1 – Overview* v1.1
[Op3] – Open IPTV Forum – *Release 1 Specification volume 2 – Media Formats* v1.1
[Op4] – Open IPTV Forum – *Release 1 Specification volume 4 – Protocols* v1.1

[Op5] – Open IPTV Forum – *Release 1 Specification volume 5 – Declaration Application Environment* v1.1
[TS] – ETSI TS 102 809 v1.1.1 "*Digital Vídeo Broadcasting (DVB); Signalling and carriage of interactive applications ans services in Hybrid broadcast / broadband environments*"

[tu1] – Tutorial javascript: www.javascriptya.com

[tu2] – Tutorial php: www.programalotu/php.com

[tu3] – Tutorial HTML+CSS: www.programalotu/html.com

[tu4] – Tutorial phpMyAdmin: www.aprendizajeweb.mx/phpmyadmin

[Xam] – *XamppServer, a Windows web development environment*:
http://www.xamppserver.com/en/

[MyA] – phpMyAdmin: http://www.phpmyadmin.net/home_page/index.php
[Ftv] – Complements Firefox. Fire HbbTV: https://addons.mozilla.org/eses/firefox/addon/firehbbtv/
[inf] – Información sobre Fire HbbTV: http://tumiptv.aw.atosorigin.com/firehbbtv/index.php

[MIT] – MIT-xperts HbbTV testsuite: http://itv.mit-xperts.com/hbbtvtest/index.php

[Ope] – Opera: http://www.operasoftware.com/products/tv-emulator

[RTVE] - RTVE: http://www.rtve.es/television/boton-rojo/

[WIK] - Wikipedia:

- http://en.wikipedia.org/wiki/Smart_TV
- http://en.wikipedia.org/wiki/Google_TV
- http://en.wikipedia.org/wiki/Ubuntu_TV
- http://en.wikipedia.org/wiki/Yahoo!_Connected_TV

[MM] – Mediamarkt: www.mediamarkt.es/mp/article/hbbtv

[app] – Aplicaciones HbbTV visitadas

http://tv-html.irt.de

http://itv.ard.de/ardepg/index.php

[VB] – Virtual box: www.virtualbox.org

[app2] – Algunas aplicaciones para SmartTV

http://www.youtube.com/watch?v=CmJDv1-AW4M
http://www.youtube.com/watch?v=ETQ7zyOmr-o
http://www.youtube.com/watch?v=TP43Ts78sBw
http://www.youtube.com/watch?v=40iwJ9BAN6M

ANEXOS

ANEXO I) EMULADORES HBBTV

1- Introducción

2- FIRE HbbTV

Descarga de addon

Prueba de aplicaciones

3- Opera HbbTV Emulator

Preparativos

Puesta en marcha

Prueba de aplicaciones

1-Introducción

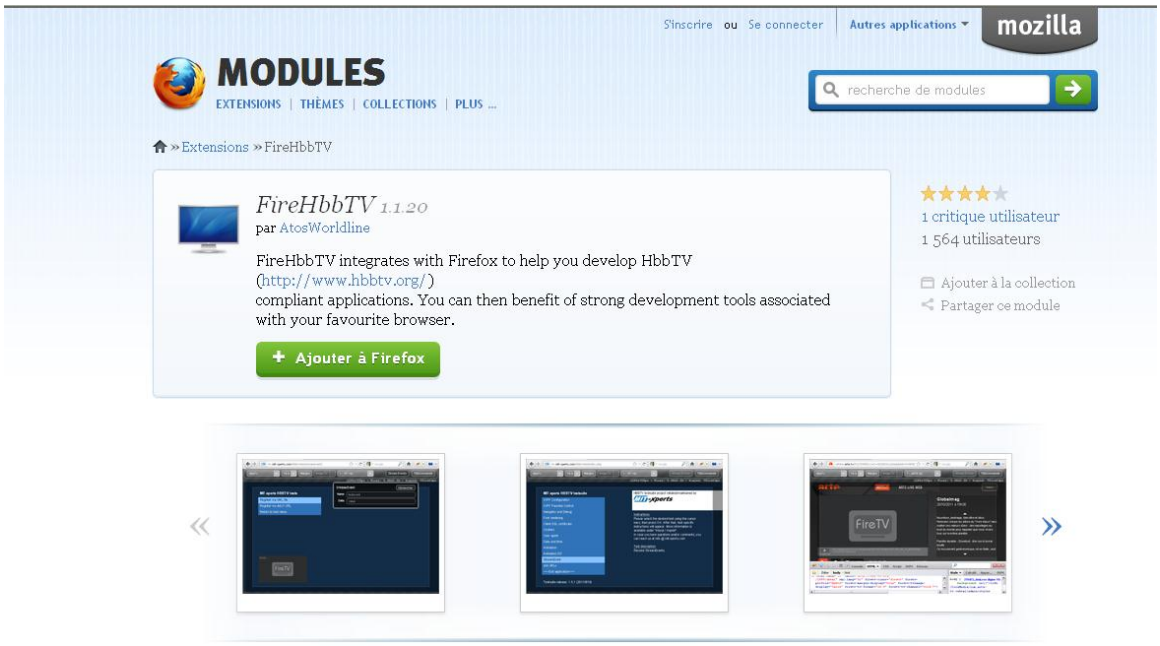
Cuando un desarrollador crea una aplicación HbbTV necesita de herramientas para poder probar si el funcionamiento de dicha aplicación es el correcto. Algunos navegadores han desarrollado ciertos paquetes que permiten visualizar aplicaciones ya desarrolladas, y además ofrecen la posibilidad de probar aplicaciones propias.

2-Fire HbbTV

FireHbbTV se integra con Firefox para poder desarrollar aplicaciones HbbTV compatibles.

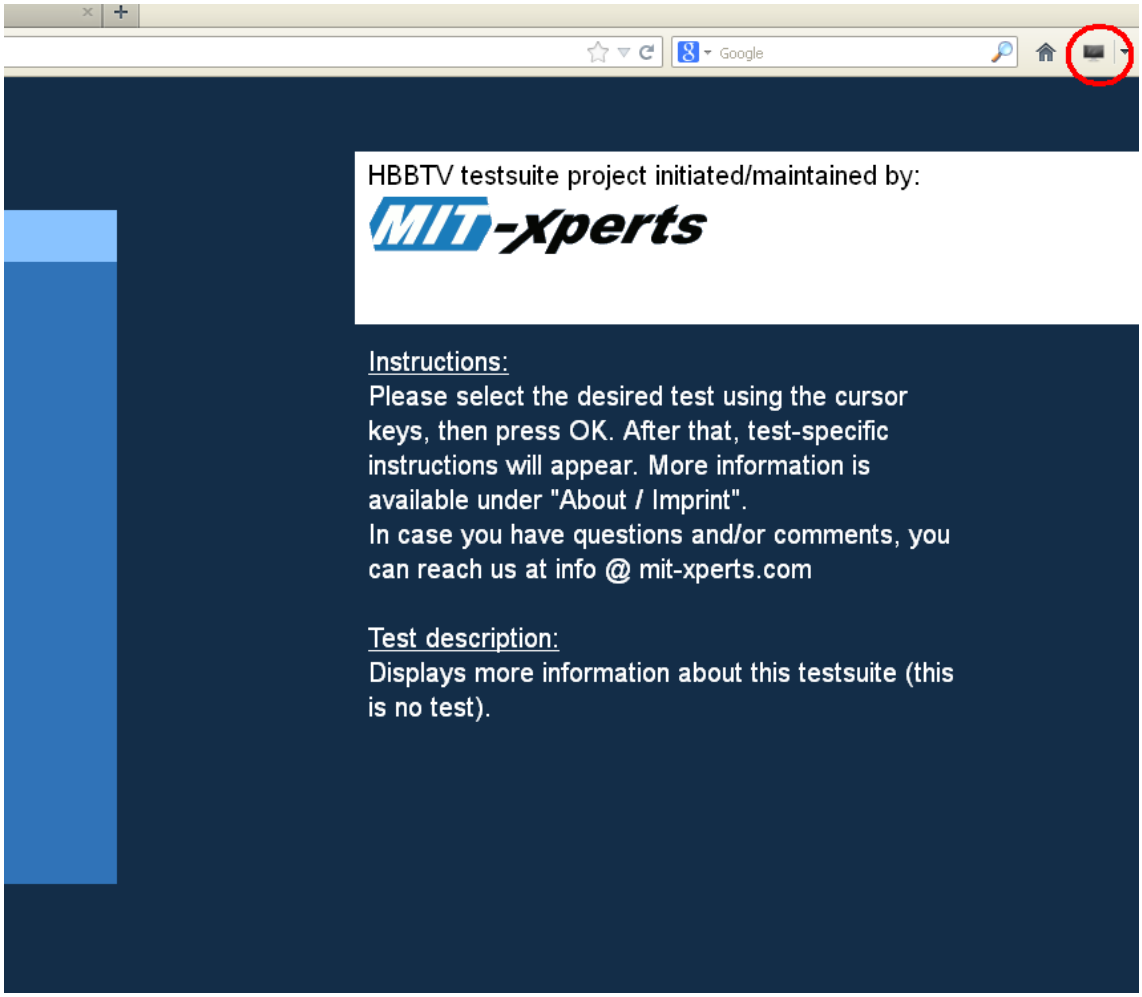
Descarga de addon

Para poder utilizar es necesario descargar el siguiente paquete que ofrece Firefox [2].



Paquete a descargar para poder utilizar Fire Hbbtv

Una vez instalado, hay que reiniciar el navegador. Al reiniciarlo aparecerá en la barra de herramientas del navegador un nuevo botón, con forma de televisión, que será el que tengamos que pulsar cuando deseemos ver las aplicaciones HbbTV.

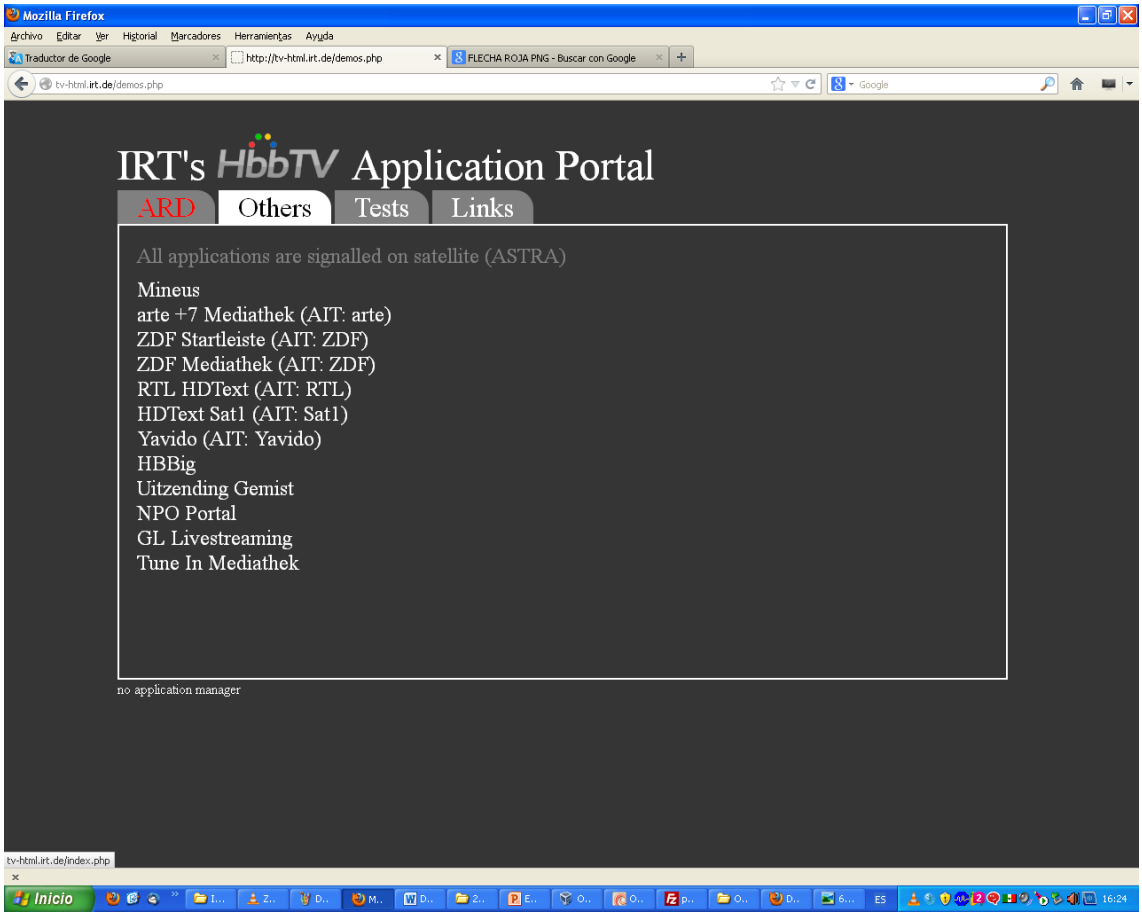


Señalado con un círculo rojo el botón a pulsar para activar el emulador HbbTV.

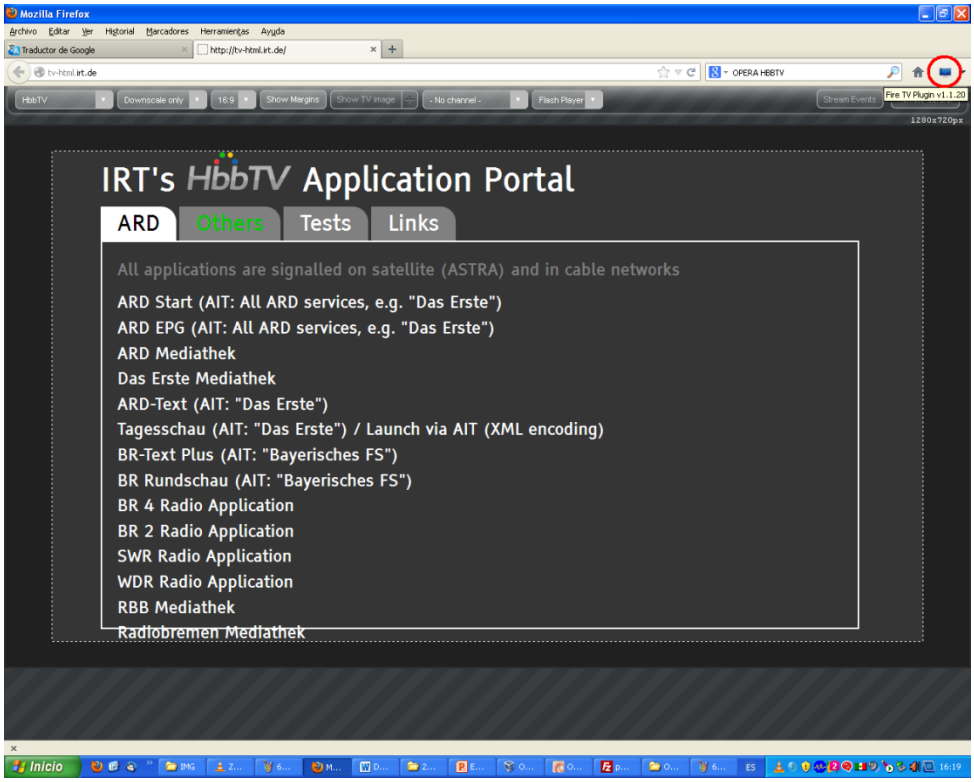
Prueba de aplicaciones

Para probar aplicaciones, lo único que hay que hacer es cargar su URL en la barra de direcciones y a continuación pulsar sobre el botón antes citado.

A continuación vemos una aplicación cargada, antes y después de haber pulsado dicho botón.

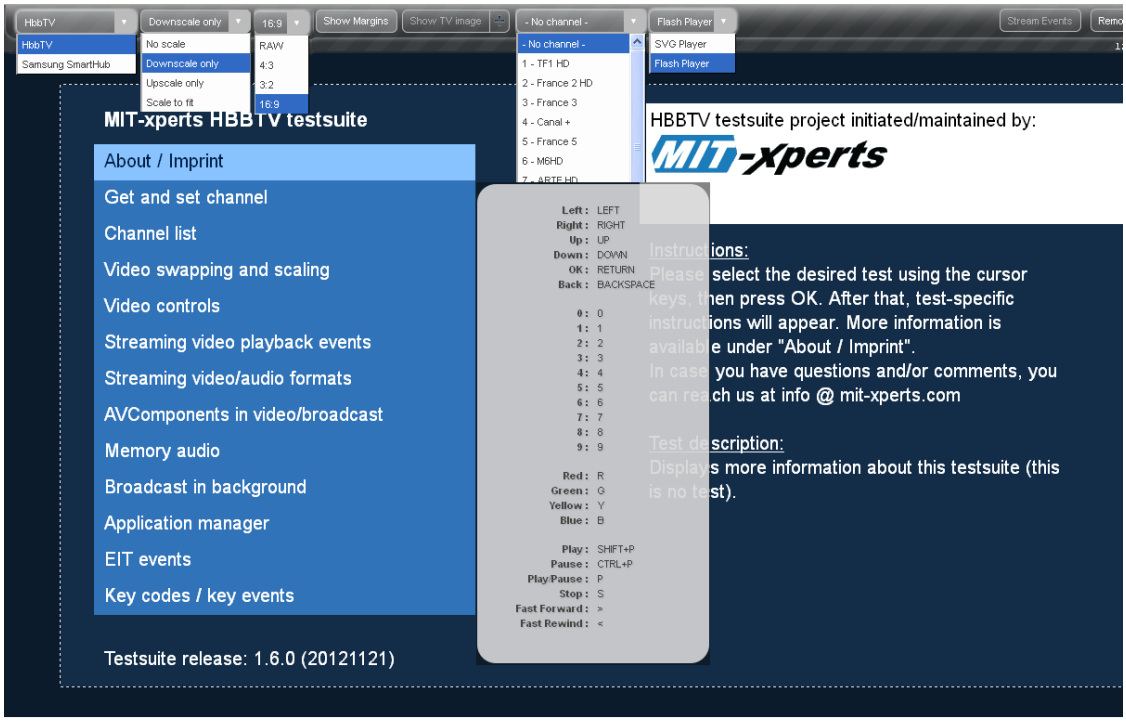


Aplicación antes de pulsar el botón HbbTV.



Aplicación ya en HbbTV.

Como se ve en la imagen, entre el cuadro de la aplicación y la barra de herramientas hay una serie de botones que nos permiten cambiar ciertos parámetros. El último de ellos es el mando de control, que indica qué botones equivalen a los de un mando de televisión.



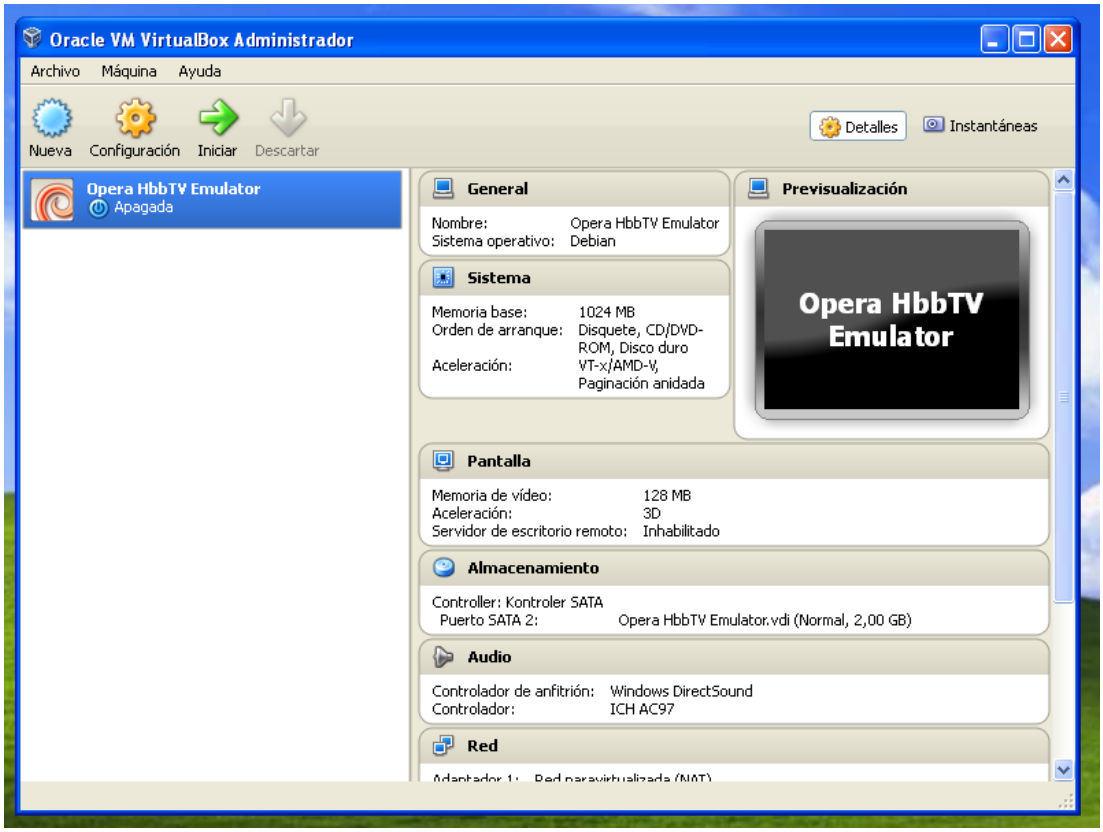
3- Opera HbbTV Emulator

Opera ofrece un paquete que permite tanto visualizar aplicaciones HbbTV vía HTTP o desde el disco duro del ordenador mediante un emulador del protocolo *object carousel*.

Para poder utilizar la versión beta de ***Opera HbbTV Emulator*** lo primero que hay que hacer es descargarse el paquete que ofrece Opera [3] así como Oracle Virtual Box [12].

Preparativos

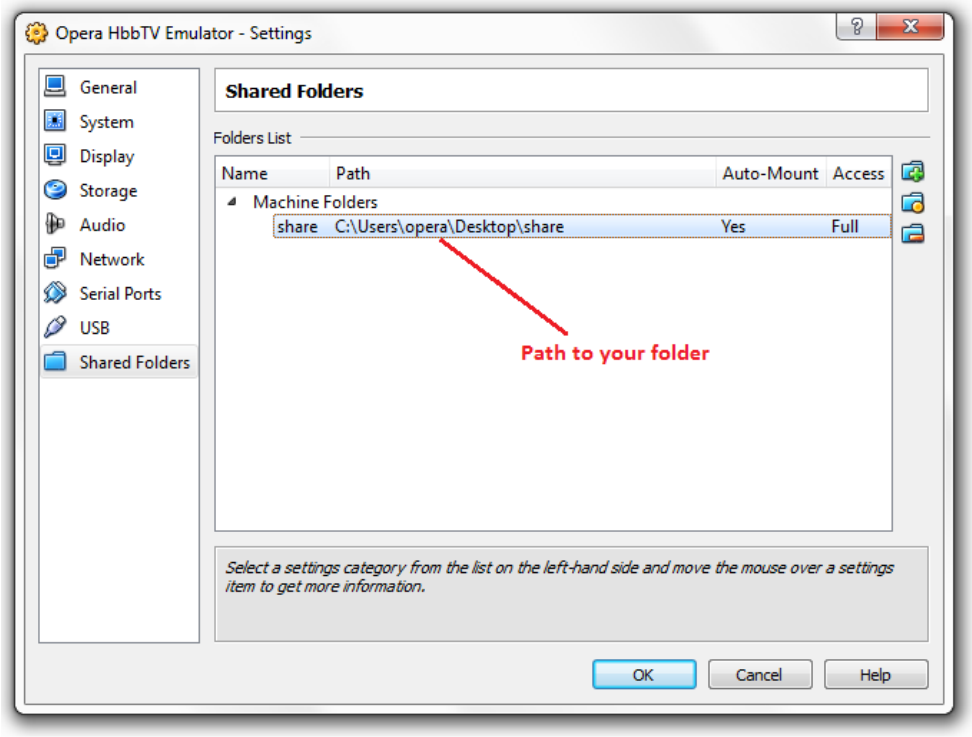
Una vez se han instalado estos dos componentes, lo primero que hay que hacer es cargar el Virtual Box de Opera.



Opera HbbTV Emulator cargada en Oracle VM.

Tras ello, el siguiente paso es pulsar el botón iniciar de tal forma que accedemos a una nueva ventana que ya será el emulador HbbTV. En esta pantalla habrá que pulsar el botón configurar, pues para poder cargar aplicaciones a la máquina virtual de Opera es necesario crear una carpeta compartida.

Opera recomienda aquí realizar dos subcarpetas, una llamada *carousel* y otra llamada *stream*. En realidad esto no es necesario pues esta versión no soporta la carga de vídeos.

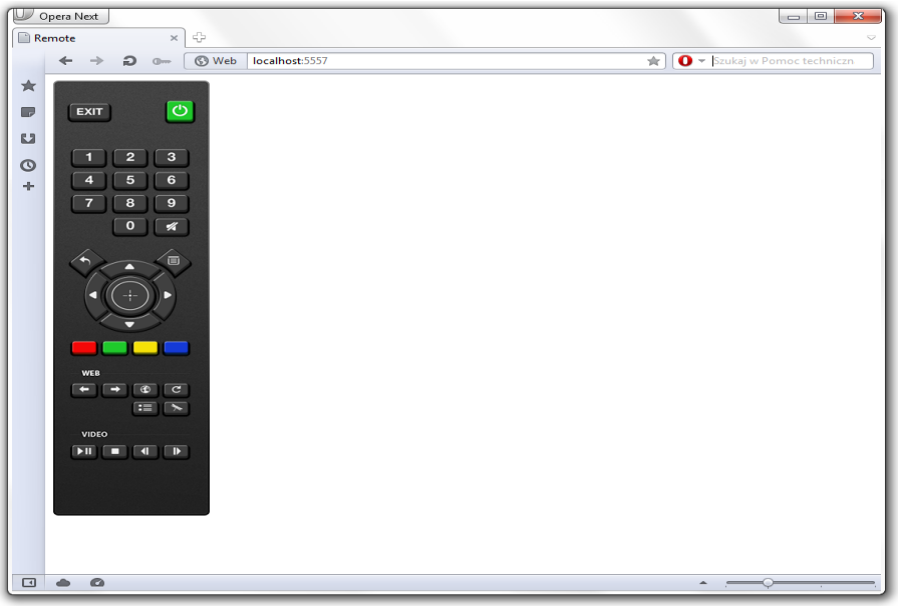


Ventana de configuración de la máquina virtual de Opera.

Puesta en marcha

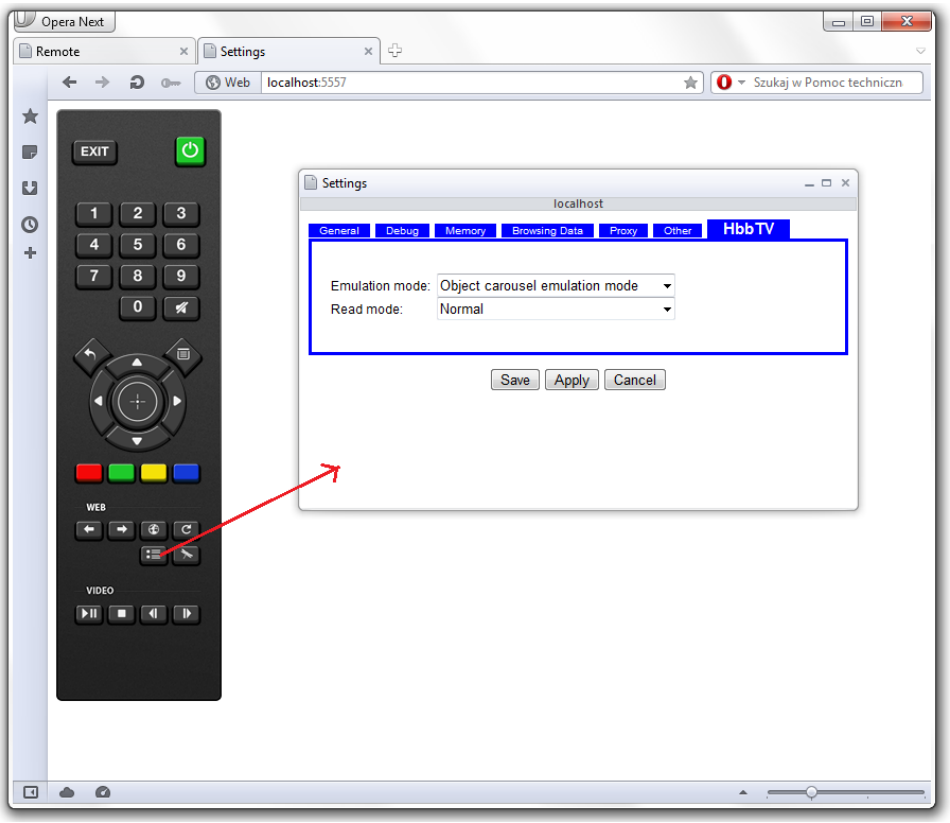
Una vez iniciada la máquina virtual, es necesario abrir un mando de control remoto basado en internet. Es una aplicación que te permitirá emular un dispositivo real de control remoto y configurar el emulador HbbTV.

Para iniciar un control remoto basado en web, hay que ir al sitio *http://localhost:5557*.



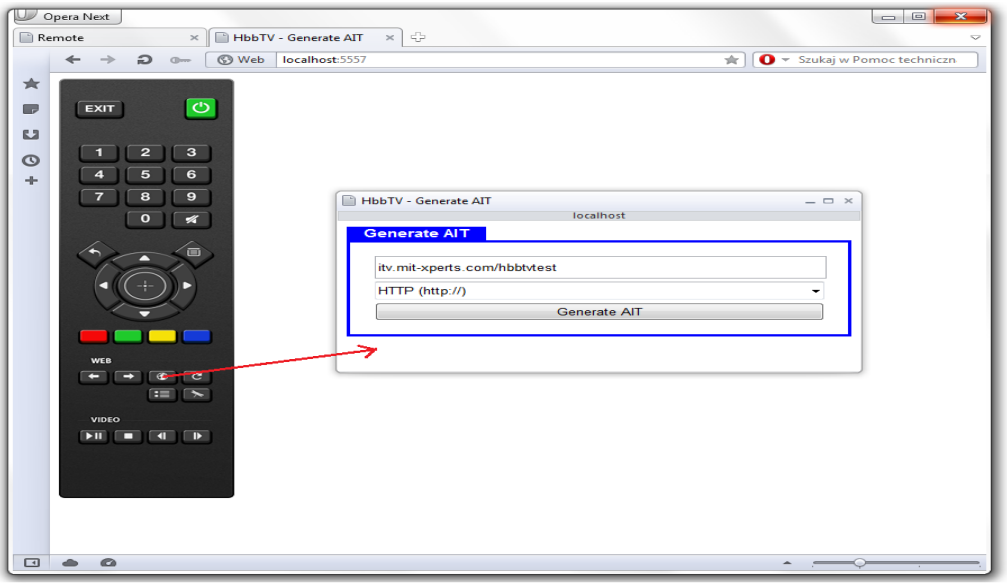
Control Remoto.

Para acceder al menú, hay que hacer clic en el botón *Configuración*. El diálogo de configuración tiene una pestaña para HbbTV.



Menú configuración.

En un escenario de *broadcast* convencional, la señalización de la aplicación se realiza a través de la tabla AIT. Pulsando el botón que se ve en la imagen siguiente se genera la tabla de la AIT.



Generador AIT.

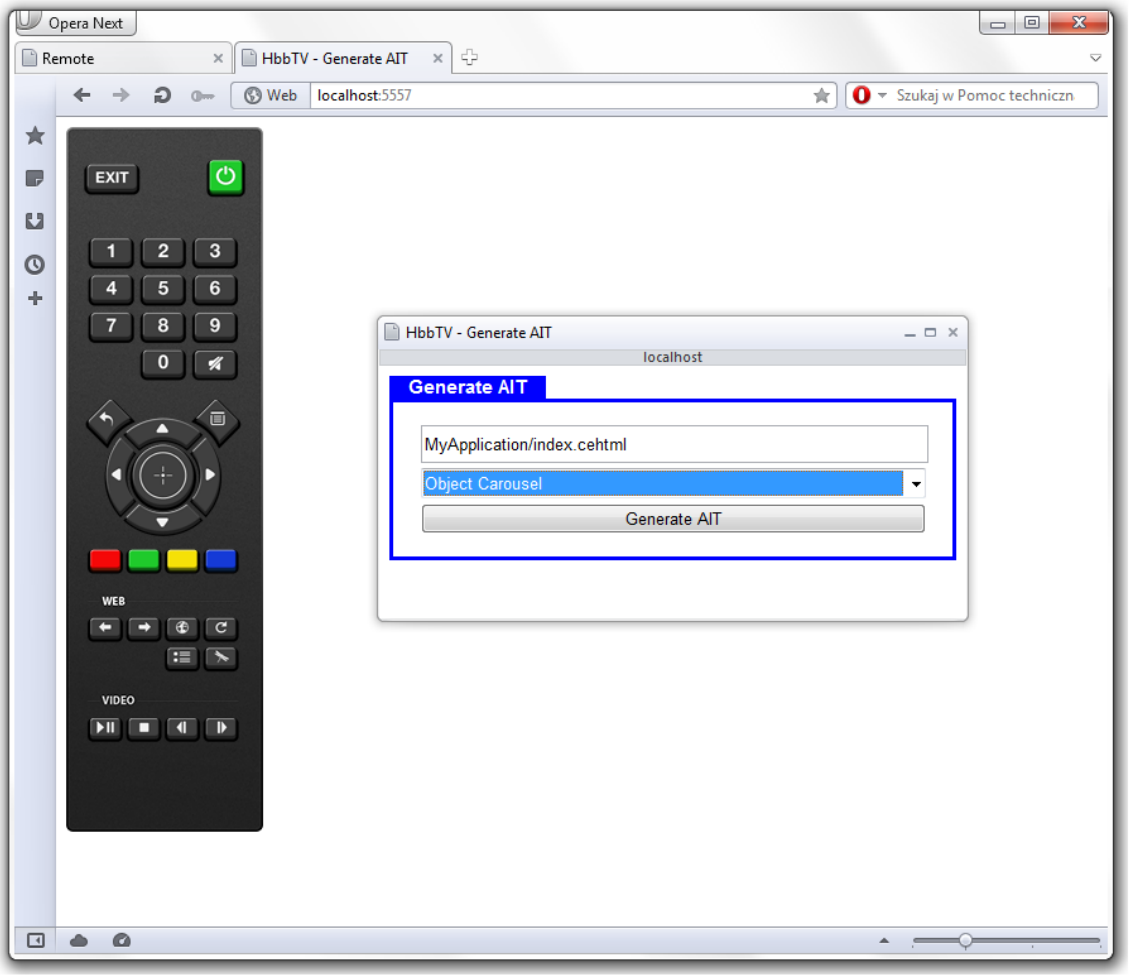
Prueba de aplicaciones

Si tenemos una aplicación HbbTV en el disco duro local, hay que copiar la carpeta que contiene dicha aplicación en el directorio *Carousel*.

Supongamos que el nombre de carpeta de la aplicación es *MyApplication* y el nombre de inicio de esta página es *index.cehtml*. Para iniciar esta aplicación, se abre la ventana para generar la AIT y se escribe el texto siguiente:

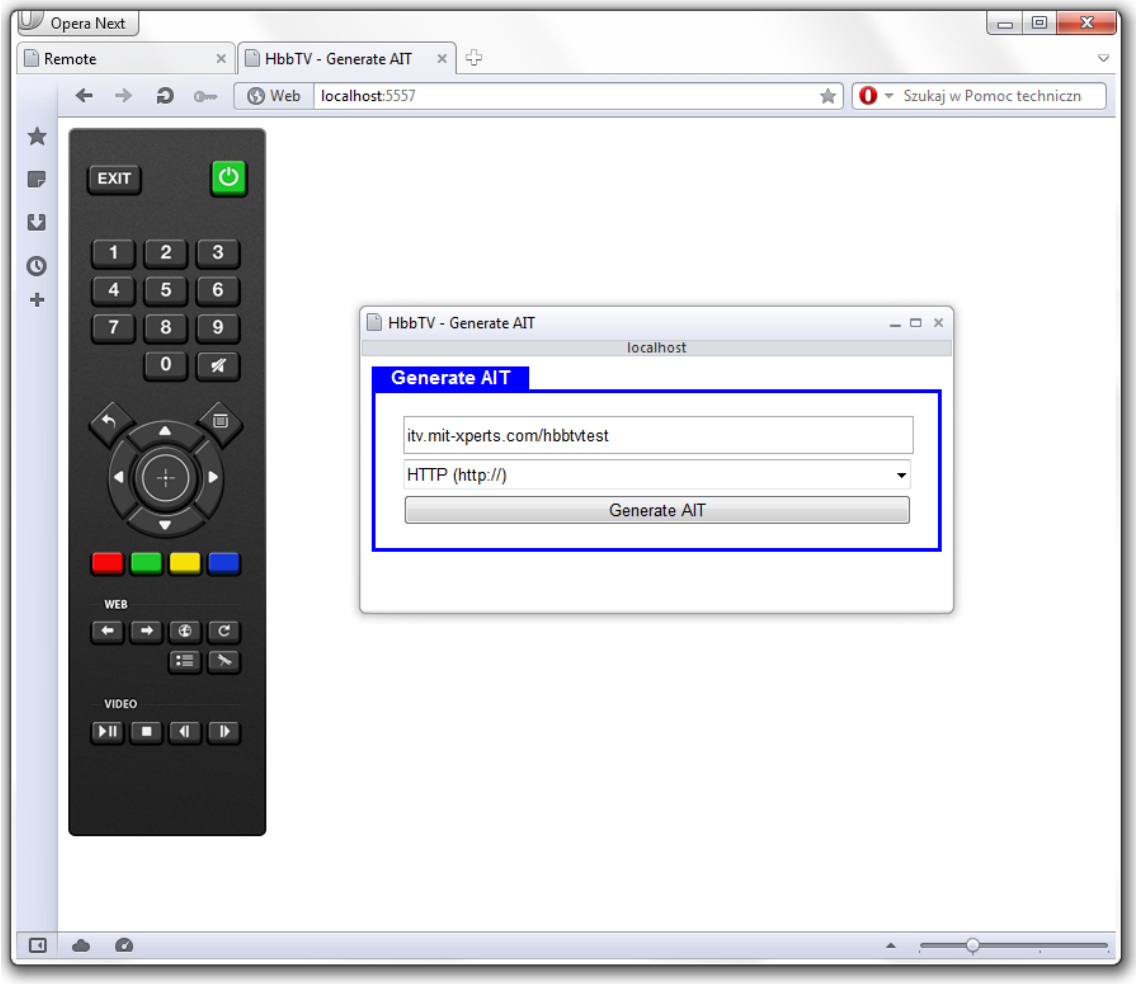
MyApplication / index.cehtml

A continuación, en el menú desplegable se elige la opción *Object Carousel* y a continuación se pulsa en *Generate AIT* .De esta forma se cargará la aplicación



Forma de cargar aplicaciones locales.

Si en caso contrario lo que queremos es probar una aplicación HbbTV que se encuentra alojada en un servidor remoto (por ejemplo itv.mit-xperts.com/hbbtvtest/index.php), se elige la opción *HTTP (http:///)* en el menú y se pulsa *Generate AIT*.



Forma de cargar aplicaciones de internet.

ANEXO II) API JAVASCRIPT IÑAKI ÚCAR

OIPF Javascript API

Iñaki Úcar

Resumen de las partes principales

- Object Factory API
  - This section defines the methods to check and create an instance of the DAE defined embedded objects within JavaScript.
  - A través del objecto global `oipfObjectFactory`
- Application Management API
  - Gestión de las aplicaciones: memoria, estado, permisos, etc.
- Configuration API
  - This section defines the interface to configuration and user settings information. Hardware configuration of the OITF is managed via an instance of the LocalSystem object. This provides access to hardware information and provides an entry point to configure the outputs and network interfaces of the OIF. Settings relating to the user interface and behaviour of the platform software are managed via an instance of the Configuration object.
- Content Download API
  - This section defines the content-on-demand download interfaces for both DRM-protected and non-DRM protected content.

Resumen de las partes principales

- Content OnDemand Metadata API
  - Gestión de catálogos de contenidos, metadatos, etc.
- Content Service Protection API
  - Digital Rights Management.
- Gateway Discovery and Control API
  - The application/oipfGatewayInfo object SHALL provide the information of the gateway and subsequently interact with the gateway (e.g. IMS Gateway, Application Gateway, CSPG-CI+ Gateway and CSPG-DTCP Gateway).
- Communication Services API
  - Servicios de mensajería instantánea, si lo soporta el cliente.
- Parental Rating and Control API
- Scheduled Recording API
  - This section describes the APIs needed to support control by a DAE application of the recording (PVR) functionality available to an OITF, including time-shift support, scheduled recording and storage of basic metadata about recorded items.

Resumen de las partes principales

- Remote Management API
  - This section defines interfaces to perform remote diagnostics and management of the device.
- Metadata API
  - This section defines the JavaScript APIs used by DAE applications for reading and searching metadata about programmes. This API does not specify whether these query operations are carried out on the OITF or whether they require communication with a server.
- Scheduled Content and Hybrid Tuner API
  - This section SHALL apply to OITFs that have indicated support for tuner control (i.e. <video_broadcast>true</video_broadcast> as defined in section 9.3.1) in their capability. It describes the video/broadcast embedded object needed to support display and control by a DAE application of scheduled content received over local tuner functionality available to an OITF, including the conveyance of the channel list to the server.
- Media Playback API
  - This section specifies several extensions to the audio object and the video object defined in section 5.7.1 of [CEA2014A].
- Miscellaneous API
- Shared Utility Classes and Features

1.1 Object Factory API: window.oipfObjectFactory

Boolean **isObjectSupported**(String mimeType)	
Visual objects, devuelven HTMLObjectElement	
createVideoBroadcastObject()	video/broadcast
createVideoMpegObject()	video/mpeg
createStatusViewObject()	application/oipfStatusView
Non-visual objects, devuelven Object	
createApplicationManagerObject()	application/oipfApplicationManager
createCapabilitiesObject()	application/oipfCapabilities
createCodManagerObject()	application/oipfCodManager
createConfigurationObject()	application/oipfConfiguration
createDownloadManagerObject()	application/oipfDownloadManager
createDownloadTriggerObject()	application/oipfDownloadTrigger
createDrmAgentObject()	application/oipfDrmAgent
createGatewayInfoObject()	application/oipfGatewayInfo
createIMSObject()	Application/oipfIMS
createMDTFObject()	application/oipfMDTF
createNotifSocketObject()	application/notifsocket
createParentalControlManagerObject()	application/oipfParentalControlManager
createRecordingSchedulerObject()	application/oipfRecordingScheduler
createRemoteManagementObject()	application/oipfRemoteManagement
createSearchManagerObject()	application/oipfSearchManager

1.2 Object Factory API: ejemplos

- Reproductor

```
var videoPlayer;
if (window.oipfObjectFactory.isObjectSupported("video/mpeg")) {
        videoPlayer = window.oipfObjectFactory.createVideoMpegObject();
        // append object to document
        document.getElementById('playerDiv').appendChild(videoPlayer);
        videoPlayer.data = "rtsp://server/barker_channel";
}
```

- Excepción

```
try {
        configuration = window.oipfObjectFactory.createConfigurationObject();
}
catch (error) {
        alert("application/oipfConfiguration object could not be created - error name: " + error.name
+ " - error message: " + error.message);
}
```

2.1 Application Management API: ApplicationManager object

Properties	
function **onLowMemory**	DOM 2
function **onApplicationLoaded**(Application appl)	DOM 2
function **onApplicationUnloaded**(Application appl)	DOM 2
Methods	
Integer **getApplicationVisualizationMode**()	
Application **getOwnerApplication**(Document document)	
ApplicationCollection **getChildApplications**(Application application)	
void **gc**()	
function **onApplicationLoadError**(Application appl)	DOM 2

2.2 Application Management API: Application

Properties	
readonly Boolean **visible**	
readonly Boolean **active**	
readonly StringCollection **permissions**	
readonly Boolean **isPrimaryReceiver**	
readonly Window **window**	
readonly ApplicationPrivateData **privateData**	
function **onApplicationActivated**	DOM 0
function **onApplicationDeactivated**	DOM 0
function **onApplicationShown**	DOM 0
function **onApplicationHidden**	DOM 0
function **onApplicationPrimaryReceiver**	DOM 0
function **onApplicationNotPrimaryReceiver**	DOM 0
function **onApplicationTopmost**	DOM 0
function **onApplicationNotTopmost**	DOM 0
function **onApplicationDestroyRequest**	DOM 0
function **onKeyPress**	DOM 0
function **onKeyUp**	DOM 0
function **onKeyDown**	DOM 0

2.2 Application Management API: Application

Methods
void **show**()
void **hide**()
void **activateInput**(Boolean gainFocus)
void **deactivateInput**()
Application **createApplication**(String uri, Boolean createChild)
void **destroyApplication**()

2.3 Application Management API: ApplicationCollection

Properties
readonly Integer **length**
Methods
Application **item**(Integer index)

2.4 Application Management API: ApplicationPrivateData

Properties
readonly Keyset **keyset**
Methods
Integer **getFreeMem**()

2.5 Application Management API: Keyset

Constants	
RED	Used to identify the VK_RED key event.
GREEN	Used to identify the VK_GREEN key event.
YELLOW	Used to identify the VK_YELLOW key event.
BLUE	Used to identify the VK_BLUE key event.
NAVIGATION	Used to identify the VK_UP, VK_DOWN, VK_LEFT, VK_RIGHT, VK_ENTER and VK_BACK key events.
VCR	Used to identify the VK_PLAY, VK_PAUSE, VK_STOP, VK_NEXT, VK_PREV, VK_FAST_FWD, VK_REWIND, VK_PLAY_PAUSE key events.
SCROLL	Used to identify the VK_PAGE_UP and VK_PAGE_DOWN key events.
INFO	Used to identify the VK_INFO key event.
NUMERIC	Used to identify the number events, 0 to 9.
ALPHA	Used to identify all alphabetic events.
OTHER	Used to indicate key events not included in one of the other constants in this class.
Properties	
readonly Integer **value**	
readonly Integer **otherKeys**[]	
readonly Integer **maximumValue**	
readonly Integer **maximumOtherKeys**[]	
Methods	
Integer **setValue**(Integer value, Integer otherKeys[])	

3.1 Configuration API: Configuration object

Properties	
readonly Configuration **configuration**	
readonly LocalSystem **localSystem**	
function **onIpAddressChange**(NetworkInterface item, String ipAddress)	DOM 2

3.2 Configuration API: Configuration

Properties
String **preferredAudioLanguage**
String **preferredSubtitleLanguage**
String **preferredUILanguage**
String **countryId**
Integer **regionId**
Integer **pvrPolicy**
Integer **pvrSaveEpisodes**
Integer **pvrSaveDays**
Integer **pvrStartPadding**
Integer **pvrEndPadding**
Methods
String **getText**(String key)
void **setText**(String key, String value)

3.3 Configuration API: LocalSystem

Properties
readonly String **deviceID**
readonly Boolean **systemReady**
readonly String **vendorName**
readonly String **modelName**
readonly String **familyName**
readonly String **softwareVersion**
readonly String **hardwareVersion**
readonly String serialNumber
readonly Boolean **pvrEnabled**
Boolean **standbyState**
Integer **volume**
Boolean **mute**
readonly AVOutputCollection **outputs**
readonly NetworkInterfaceCollection **networkInterfaces**
readonly Integer **pvrSupport**
readonly Boolean **ciplusEnabled**
readonly Integer **releaseVersion**
readonly Integer **majorVersion**
readonly Integer **minorVersion**
readonly String **oipfProfile**

3.3 Configuration API: LocalSystem

Methods
Boolean **setScreenSize**(Integer width, Integer height)
Integer **setPvrSupport**(Integer state)

3.4 Configuration API: NetworkInterface

Properties
readonly String **ipAddress**
readonly String **macAddress**
readonly Boolean **connected**
Boolean **enabled**

3.5 Configuration API: NetworkInterfaceCollection

Properties
readonly Integer **length**
Methods
NetworkInterface **item**(Integer index)

3.6 Configuration API: AVOutput

Properties
readonly String **name**
readonly String **type**
Boolean **enabled**
Boolean **subtitleEnabled**
String **videoMode**
String **digitalAudioMode**
String **audioRange**
String **hdVideoFormat**
String **tvAspectRatio**
readonly StringCollection **supportedVideoModes**
readonly StringCollection **supportedDigitalAudioModes**
readonly StringCollection **supportedAudioRanges**
readonly StringCollection **supportedHdVideoFormats**
readonly StringCollection **supportedAspectRatios**

3.7 Configuration API: AVOutputCollection

Properties
readonly Integer **length**
Methods
AVOutput **item**(Integer index)

4.1 Content Download API: DownloadTrigger object

Methods
String **registerDownload**(String contentAccessDownloadDescriptor, Date downloadStart)
String **registerDownloadURL**(String URL, String contentType, Date downloadStart)
Integer **checkDownloadPossible**(Integer sizeInBytes)
String **registerDownloadFromCRID**(String CRID, String IMI, Date downloadStart)

4.2 Content Download API: DownloadManager object

Properties	
function **onDownloadStateChange**(Download item, Integer state, Integer reason)	DOM 2
readonly DiscInfo **discInfo**	
Methods	
Boolean **pause**(Download download)	
Boolean **resume**(Download download)	
Boolean **remove**(Download download)	
DownloadCollection **getDownloads**(String id)	
DownloadCollection **createFilteredList**(Boolean currentDomain, Integer states)	

4.3 Content Download API: Download

Properties
readonly Integer **totalSize**
readonly Integer **state**
readonly Integer **amountDownloaded**
readonly String **name**
readonly String **id**
readonly String **contentURL**
readonly String **description**
readonly ParentalRatingCollection **parentalRatings**
readonly DRMControlInfoCollection **drmControl**
readonly Date **startTime**
readonly Integer **timeElapsed**
readonly Integer **timeRemaining**
readonly String **transferType**
readonly String **originSite**
readonly String **originSiteName**
readonly String **contentID**
readonly String **iconURL**

4.4 Content Download API: DownloadCollection

Properties
readonly Integer **length**
Methods
Download **item**(Integer index)

4.5 Content Download API: DRMControlInfoCollection

Properties
readonly Integer **length**
Methods
DRMControlInformation **item**(Integer index)

4.6 Content Download API: DRMControlInformation

Properties
readonly String **drmType**
readonly String **rightsIssuerURL**
readonly String **silentRightsURL**
readonly String **silentRightsURL**
readonly String **previewRightsURL**
readonly String **drmPrivateData**
readonly Boolean **doNotRecord**
readonly Boolean **doNotTimeShift**

5.1 Content OnDemand Metadata API: CodManager object

Properties	
readonly ContentCatalogueCollection **catalogues**	
function **onContentCatalogueEvent**(Integer action)	DOM 2
function **onContentAction**(Integer action, Integer result, Object item, ContentCatalogue catalogue)	DOM 2

5.2 Content OnDemand Metadata API: CatalogueCollection

Properties
readonly Integer **length**
Methods
ContentCatalogue **item**(Integer index)

5.3 Content OnDemand Metadata API: ContentCatalogue

Properties
readonly String **name**
readonly CODFolder **rootFolder**
Methods
CODFolder **getPurchaseHistory**()

5.4 Content OnDemand Metadata API: ContentCatalogueEvent

5.5 Content OnDemand Metadata API: CODFolder

Properties
readonly Integer **type**
readonly String **uri**
readonly String **name**
readonly String **description**
readonly String **thumbnailUri**
readonly Integer **length**
readonly Integer **currentPage**
readonly Integer **pageSize**
readonly Integer **totalSize**
Methods
Object **item**(Integer index)
void **getPage**(Integer page, Integer pageSize)
void **abort**()

5.6 Content OnDemand Metadata API: CODAsset

Properties
readonly Integer **type**
readonly String **uri**
readonly String **name**
readonly String **description**
readonly StringCollection **genres**
readonly ParentalRatingCollection **parentalRatings**
readonly Boolean **blocked**
readonly Boolean **locked**
readonly String **thumbnailUri**
readonly String **price**
readonly Integer **rentalPeriod**
readonly Integer **playCount**
readonly Integer **duration**
readonly String **previewUri**
readonly BookmarkCollection **bookmarks**
Methods
Boolean **isReady**()

5.7 Content OnDemand Metadata API: CODService

Properties
readonly Integer **length**
readonly Integer **currentPage**
readonly Integer **pageSize**
readonly Integer **totalSize**
readonly Integer **type**
readonly String **uri**
readonly String **name**
readonly String **description**
readonly String **thumbnailUri**
readonly String **previewUri**
Methods
Boolean **isReady**()
Object **item**(Integer index)
void **getPage**(Integer page, Integer pageSize)
void **abort**()

6.1 Content Service Protection API: DRMAgent object

Properties	
function **onDRMMessageResult**(String msgID, String resultMsg, Integer resultCode)	DOM 2
function **onDRMSystemMessage**(String msg, String DRMSystemID)	DOM 2
Methods	
String **sendDRMMessage**(String msgType, String msg, String DRMSystemID)	

7.1 Gateway Discovery/Control API: GatewayInfo object

Properties	
readonly Boolean **isIGDiscovered**	
readonly Boolean **isAGDiscovered**	
readonly Boolean **isCSPGCIPlusDiscovered**	
readonly Boolean **isCSPGDTCPDiscovered**	
readonly String **igURL**	
readonly String **agURL**	
readonly String **cspgDTCPURL**	
Integer **interval**	
function **onDiscoverIG**	DOM 2
function **onDiscoverAG**	
function **onDiscoverCSPGDTCP**	DOM 2
readonly Boolean **isIGSupported**	
readonly Boolean **isAGSupported**	
readonly Boolean **isCSPGCIPlusSupported**	
readonly Boolean **isCSPGDTCPSupported**	
function **onDiscoverCSPGCIPlus**	DOM 2
readonly StringCollection **CSPGCIPlusDRMType**	
Methods	
Boolean **isIGSupportedMethod**(String MethodName)	

8.1 Communication ServicesAPI: CommunicationServices object

Constants	
STATE_REGISTERED	
STATE_REGISTERED_SUBSCRIPTION_PENDING	
STATE_REGISTERED_SUBSCRIPTION_ACTIVE	
STATE_DEREGISTERED	
STATE_FAILURE	
Properties	
function **onNotification**(String responseHeaders, String msgText, Document msgXML)	DOM 2
function **onNotificationResult**(Integer resultMsg)	DOM 2
function **onRegistrationContextUpdate**(String user, Integer state, Integer errorCode)	DOM 2
readonly UserData **currentUser**	
Methods	
UserDataCollection **getRegisteredUsers**()	
Void **registerUser**(String userId, String pin)	
void **deRegisterUser**(String userId)	
UserDataCollection **getAllUsers**()	
Boolean **setUser**(String userId)	
void **subscribeNotification**(FeatureTagCollection featureTagCollection, Boolean performUserRegistration)	
void **unsubscribeNotification**()	

8.1 Communication ServicesAPI: CommunicationServices object

Properties (extension)	
function **onIncomingMessage**(String fromURI, String msg, Integer cid)	DOM 2
function **onContactStatusChange**(String remoteURI, Integer state)	DOM 2
function **onNewWatcher**(String remoteURI)	DOM 2
Methods (extension)	
Integer **openChatSession**(String toURI)	
void **sendMessageInSession**(Integer cid, String msg)	
void **closeChatSession**(Integer cid)	
void **sendMessage**(String toURI, String msg)	
void **setStatus**(Integer state)	
void **subscribeToStatus**(String remoteURI)	
ContactCollection **getContacts**()	
void **allowContact**(String remoteURI)	
void **blockContact**(String remoteURI)	
Boolean **createContactList**(String contactListUri, ContactCollection contacts)	
ContactCollection **getContacts**(String contactListUri)	
Boolean **addToContactList**(String contactListUri, Contact member)	
Boolean **removeFromContactList**(String contactListUri, Contact member)	
Boolean **deleteContactList**(String contactListUri)	
void **allowAllContacts**(String domain)	
void **blockAllContacts**(String domain)	

8.2 Communication ServicesAPI: UserData

Properties
readonly String **userId**
readonly FeatureTagCollection **featureTags**
readonly String **friendlyName**

8.3 Communication ServicesAPI: UserDataCollection

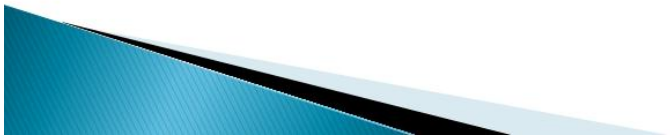
Properties
readonly Integer **length**
Methods
UserData **item**(Integer index)

8.4 Communication ServicesAPI: FeatureTag

Properties
readonly String **featureTag**

8.5 Communication ServicesAPI: FeatureTagCollection

Properties
readonly Integer **length**
Methods
FeatureTag **item**(Integer index)



8.6 Communication ServicesAPI: Contact

Properties
String **contactId**
String **friendlyName**

8.7 Communication ServicesAPI: ContactCollection

Properties
readonly Integer **length**
Methods
Contact **item**(Integer index)
Boolean **remove**(String contactId)
Boolean **add**(Contact contact)

9.1 Parental Control API: ParentalControlManager object

Properties
readonly ParentalRatingSchemeCollection **parentalRatingSchemes**
readonly Boolean **isPINEntryLocked**
Methods
Integer **setParentalControlStatus**(String pcPIN, Boolean enable)
Boolean **getParentalControlStatus**()
Boolean **getBlockUnrated**()
Integer **setParentalControlPIN**(String oldPcPIN, String newPcPIN)
Integer **unlockWithParentalControlPIN**(String pcPIN, Object target)
Integer **verifyParentalControlPIN**(String pcPIN)
Integer **setBlockUnrated**(String pcPIN, Boolean block)

9.2 Parental Control API: ParentalRatingScheme

Properties
readonly String **name**
readonly Integer **length**
readonly ParentalRating **threshold**
Methods
Integer **indexOf**(String ratingValue)
String **item**(Integer index)
String **iconUri**(Integer index)

9.3 Parental Control API: ParentalRatingSchemeCollection

Properties
readonly Integer **length**
Methods
ParentalRatingScheme **item**(Integer index)
ParentalRatingScheme **addParentalRatingScheme**(String name, String values)
ParentalRatingScheme **getParentalRatingScheme**(String name)

9.4 Parental Control API: ParentalRating

Properties
readonly String **name**
readonly String **scheme**
readonly Integer **value**
readonly Integer **labels**
readonly String **region**

9.5 Parental Control API: ParentalRatingCollection

Properties
readonly Integer **length**
Methods
ParentalRating **item**(Integer index)
void **addParentalRating**(String scheme, String name, Integer value, Integer labels, String region)

10.1 Scheduled Recording API: RecordingScheduler object

Methods
ScheduledRecording **record**(Programme programme)
ScheduledRecording **recordAt**(Integer startTime, Integer duration, Integer repeatDays, String channelID)
ScheduledRecordingCollection **getScheduledRecordings**()
ChannelConfig **getChannelConfig**()
void **remove**(ScheduledRecording recording)
Programme **createProgrammeObject**()

10.2 Scheduled Recording API: ScheduledRecording

Constants
RECORDING_SCHEDULED
RECORDING_REC_STARTED
RECORDING_REC_COMPLETED
RECORDING_REC_PARTIALLY_COMPLETED
RECORDING_ERROR
ERROR_REC_RESOURCE_LIMITATION
ERROR_INSUFFICIENT_STORAGE
ERROR_REC_UNKNOWN
ID_TVA_CRID
ID_DVB_EVENT

10.2 Scheduled Recording API: ScheduledRecording

Properties
readonly Integer **state**
readonly Integer **error**
readonly String **scheduleID**
Integer **startPadding**
Integer **endPadding**
readonly Integer **repeatDays**
String **name**
String **longName**
String **description**
String **longDescription**
readonly Integer **startTime**
readonly Integer **duration**
readonly Channel **channel**
readonly Boolean **isManual**
readonly String **programmeID**
readonly Integer **programmeIDType**
readonly Integer **episode**
readonly Integer **totalEpisodes**
readonly ParentalRatingCollection **parentalRatings**

10.3 Scheduled Recording API: ScheduledRecordingCollection

Properties	
readonly Integer **length**	
Methods	
ScheduledRecording **item**(Integer index)	

10.4 Scheduled Recording API: RecordingScheduler object

Properties (extension)	
readonly ScheduledRecordingCollection **recordings**	
readonly DiscInfo **discInfo**	
function **onPVREvent**(Integer state, ScheduledRecording recording)	DOM 2
Methods (extension)	
Recording **getRecording**(String id)	
void **stop**(Recording recording)	
void **refresh**()	
Boolean **update**(String id, Integer startTime, Integer duration, Integer repeatDays)	

10.5 Scheduled Recording API: Recording

Properties
readonly String **id**
Boolean **doNotDelete**
Integer **saveDays**
Integer **saveEpisodes**
readonly Boolean **blocked**
readonly Integer **showType**
readonly Boolean **subtitles**
readonly StringCollection **subtitleLanguages**
readonly Boolean **isHD**
readonly Integer **audioType**
readonly Boolean **isMultilingual**
readonly StringCollection **audioLanguages**
readonly StringCollection **genres**
readonly Integer **recordingStartTime**
readonly Integer **recordingDuration**
readonly BookmarkCollection **bookmarks**
readonly Boolean **locked**

10.6 Scheduled Recording API: RecordingCollection

10.7 Scheduled Recording API: PVREventc

10.8 Scheduled Recording API: Bookmark

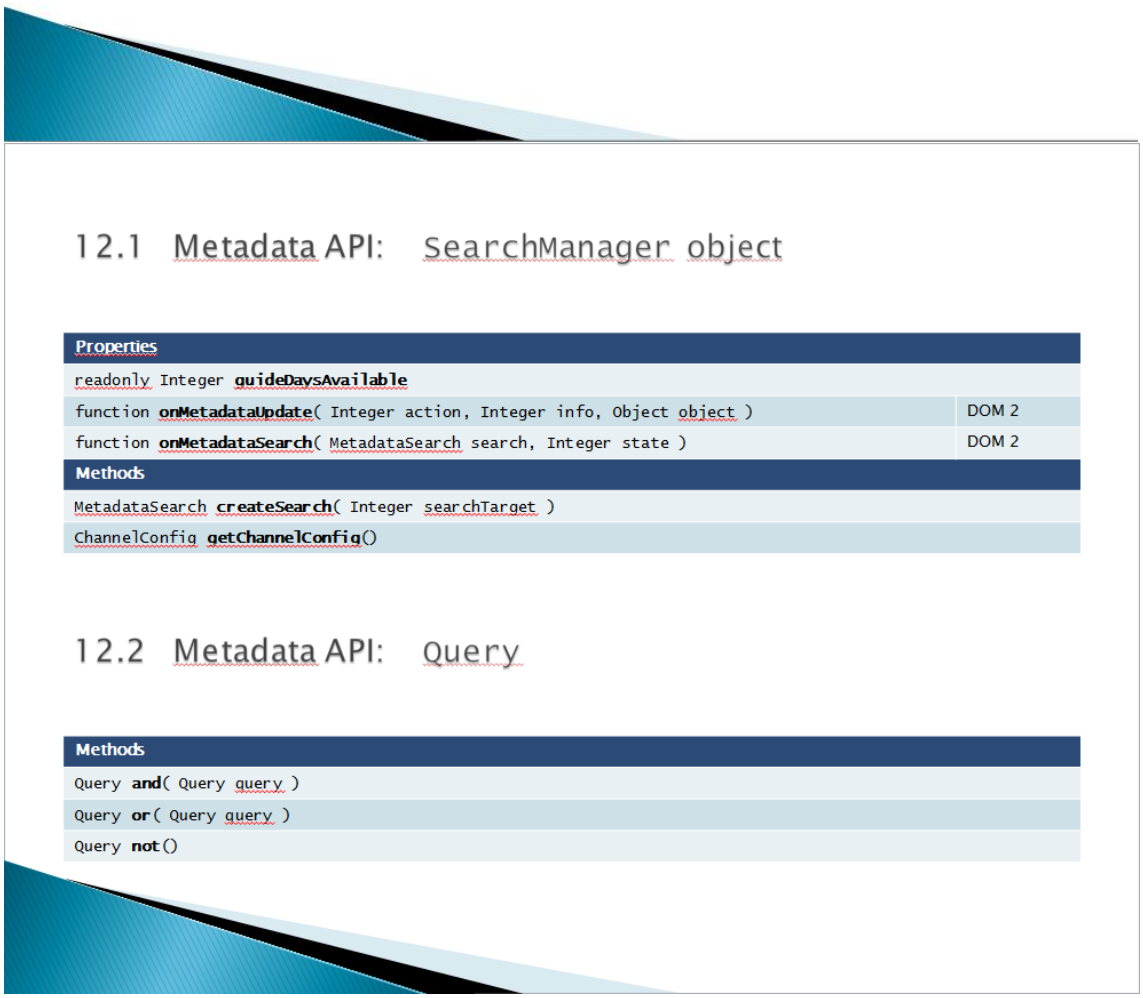
Properties
readonly Integer **time**
readonly String **name**

10.9 Scheduled Recording API: BookmarkCollection

Properties
readonly Integer **length**
Methods
Bookmark **item**(Integer index)
Bookmark **addBookmark**(Integer time, String name)
void **removeBookmark**(Bookmark bookmark)

11.1 Remote Management API: RemoteManagement object

Properties
readonly String **vendorName**
readonly String **modelName**
readonly String **softwareVersion**
readonly String **hardwareVersion**
readonly String **familyName**
Methods
String **getParameter**(String parameterName)
String **setParameter**(String parameterName, String value)
Integer **triggerSoftwareUpdate**(String token)



12.1 Metadata API: SearchManager object

Properties	
readonly Integer **guideDaysAvailable**	
function **onMetadataUpdate**(Integer action, Integer info, Object object)	DOM 2
function **onMetadataSearch**(MetadataSearch search, Integer state)	DOM 2
Methods	
MetadataSearch **createSearch**(Integer searchTarget)	
ChannelConfig **getChannelConfig**()	

12.2 Metadata API: Query

Methods
Query **and**(Query query)
Query **or**(Query query)
Query **not**()

12.3 Metadata API: MetadataSearch

Properties
readonly Integer **searchTarget**
readonly SearchResults **result**
Methods
void **setQuery**(Query query)
void **addRatingConstraint**(ParentalRatingScheme scheme, Integer threshold)
void **addCurrentRatingConstraint**()
void **addChannelConstraint**(ChannelList channels)
void **addChannelConstraint**(Channel channel)
void **orderBy**(String field, Boolean ascending)
Query **createQuery**(String field, Integer comparison, String value)
void **findProgrammesFromStream**(Channel channel, Integer startTime)

12.4 Metadata API: SearchResults

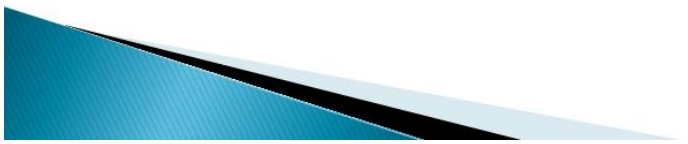
Properties
readonly Integer **length**
readonly Integer **offset**
readonly Integer **totalSize**
Methods
Object **item**(Integer index)
Boolean **getResults**(Integer offset, Integer count)
void **abort**()

12.5 Metadata API: MetadataSearchEvent

12.6 Metadata API: MetadataUpdateEvent

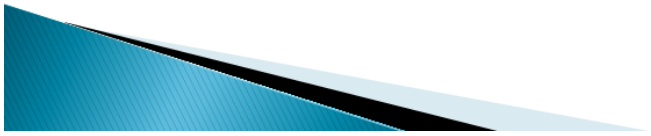
13.1 Scheduled Content API: video/broadcast object

Properties	
Integer **width**	
Integer **height**	
readonly Boolean **fullScreen**	
function **onChannelChangeError**(Channel channel, Number errorState)	DOM 2
Integer **playState**	
function **onPlayStateChange**(Number state, Number error)	DOM 2
function **onChannelChangeSucceeded**(Channel channel)	DOM 2
function **onFullScreenChange**	DOM 2
function **onfocus**	DOM 2
function **onblur**	DOM 2
String **data**	



13.1 Scheduled Content API: video/broadcast object

Method
ChannelConfig **getChannelConfig**()
void **bindToCurrentChannel**()
Channel **createChannelObject**(Integer idType, String dsd, Integer sid)
Channel **createChannelObject**(Integer idType, Integer onid, Integer tsid, Integer sid, Integer sourceID, String ipBroadcastID)
void **setChannel**(Channel channel, Boolean trickplay, String contentAccessDescriptorURL)
void **prevChannel**()
void **nextChannel**()
void **setFullScreen**(Boolean fullscreen)
Boolean **setVolume**(Integer volume)
Integer **getVolume**()
void **release**()
void **stop**()



13.1 Scheduled Content API: video/broadcast object

Properties (extension)	
function **onPlaySpeedChanged**(Number speed)	DOM 2
function **onPlayPositionChanged**(Integer position)	DOM 2
readonly Integer **playbackOffset**	
readonly Integer **maxOffset**	
readonly Integer **recordingState**	
function **onRecordingEvent**(Integer state, Integer error, String recordingId)	DOM 2
readonly Integer **playPosition**	
readonly Number **playSpeed**	
readonly Number **playSpeeds**[ ]	
readonly ProgrammeCollection **programmes**	
function **onProgrammesChanged**	DOM 2
function **onParentalRatingChange**(String contentID, ParentalRatingCollection ratings, String DRMSystemID, Boolean blocked)	DOM 2
function **onParentalRatingError**(String contentID, ParentalRatingCollection ratings, String DRMSystemID)	DOM 2
function **onDRMRightsError**(Integer errorState, String contentID, String DRMSystemID, String rightsIssuerURL)	DOM 2
readonly Channel **currentChannel**	
ChannelList **createChannelList**(String bdr)	

13.1 Scheduled Content API: video/broadcast object

Constants (extension)
POSITION_START
POSITION_CURRENT
POSITION_END
Methods (extension)
String **recordNow**(Integer duration)
void **stopRecording**()
Boolean **pause**()
Boolean **resume**()
Boolean **setSpeed**(Number speed)
Boolean **seek**(Integer offset, Integer reference)
Boolean **stopTimeshift**()
void **setChannel**(Channel channel, Boolean trickplay, String contentAccessDescriptorURL, Integer offset)

13.2 Scheduled Content API: ChannelConfig

Properties	
readonly ChannelList **channelList**	
readonly FavouriteListCollection **favouriteLists**	
readonly String **currentFavouriteList**	
function **onChannelListUpdate**	DOM 2
Methods	
ChannelList **createFilteredList**(Boolean blocked, Boolean favourite, Boolean hidden, String favouriteListID)	

13.3 Scheduled Content API: ChannelList

Properties
readonly Integer **length**
Methods
Channel **item**(Integer index)
Channel **getChannel**(String channelID)
Channel **getChannelByTriplet**(Integer onid, Integer tsid, Integer sid)
Channel **getChannelBySourceID**(Integer sourceID)

13.4 Scheduled Content API: Channel

Constants
TYPE_TV
TYPE_RADIO
TYPE_HBBTV_DATA
ID_ANALOG
ID_DVB_C
ID_DVB_S
ID_DVB_T
ID_DVB_SI_DIRECT
ID_DVB_C2
ID_DVB_S2
ID_DVB_T2
ID_ISDB_C
ID_ISDB_S
ID_ISDB_T
ID_ATSC_T
ID_IPTV_SDS
ID_IPTV_URI

13.4 Scheduled Content API: Channel

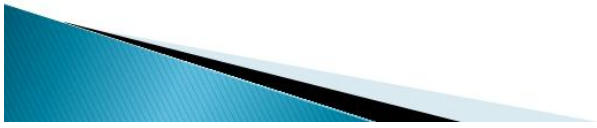
Properties
readonly Integer **channelType**
readonly Integer **idType**
readonly String **ccid**
readonly String **tunerID**
readonly Integer **onid**
readonly Integer **nid**
readonly Integer **tsid**
readonly Integer **sid**
readonly Integer **sourceID**
readonly Integer **freq**
readonly Integer **cni**
readonly String **name**
readonly Integer **majorChannel**
readonly Integer **minorChannel**
readonly String **dsd**
readonly Boolean **favourite**
readonly StringCollection **favIDs**
readonly Boolean **locked**
readonly Boolean **manualBlock**

13.4 Scheduled Content API: Channel

Properties
readonly String **ipBroadcastID**
readonly Integer **channelMaxBitRate**
readonly Integer **channelTTR**
readonly Boolean **recordable**
Properties (extension)
readonly String **longName**
readonly String **description**
readonly Boolean **authorised**
readonly StringCollection **genre**
Boolean **hidden**
string **logoURL**
Methods (extension)
String **getField**(String fieldId)
String **getLogo**(Integer width, Integer height)

13.5 Scheduled Content API: FavouriteListCollection

Properties
readonly Integer **length**
Methods
FavouriteList **getFavouriteList**(String favID)
FavouriteList **item**(Integer index)
Methods (extension)
Integer **createFavouriteList**()
Boolean **remove**(Integer index)
Boolean **commit**()



13.6 Scheduled Content API: FavouriteList

Properties
readonly String **favID**
readonly String **name**
readonly Integer **length**
Methods
Channel **item**(Integer index)
Channel **getChannel**(String channelID)
Channel **getChannelByTriplet**(Integer onid, Integer tsid, Integer sid)
Channel **getChannelBySourceID**(Integer sourceID)
Methods (extension)
Boolean **insertBefore**(Integer index, String ccid)
Boolean **remove**(Integer index)
Boolean **commit**()

14.1 Media Playback API: CEA 2014 A/V Control object

Properties (extension)	
function **onPlaySpeedChanged**(Number speed)	DOM 2
function **onPlayPositionChanged**(Integer position)	DOM 2
readonly Number **playSpeeds**[ ]	
readonly String **oitfSourceIPAddress**	
readonly String **oitfSourcePortAddress**	
Boolean **oitfNoRTSPSessionControl**	
String **oitfRTSPSessionId**	
function **onParentalRatingChange**(String contentID, ParentalRatingCollection ratings, String DRMSystemID, Boolean blocked)	DOM 2
function **onParentalRatingError**(String contentID, ParentalRatingCollection ratings, String DRMSystemID)	DOM 2
function **onDRMRightsError**(Integer errorState, String contentID, String DRMSystemID, String rightsIssuerURL)	DOM 2
function **onReadyToPlay**	DOM 2
Boolean **readyToPlay**	
Methods (extension)	
Boolean **setSource**(String id)	
Integer **getAvailablePlayTime**(Boolean fromPlayPosition)	
Boolean **setBufferingStrategy**(String name)	

15.1 Miscellaneous API: MDTF object

Properties	
function **onFLUTEListenerResult**(String multicastAddress, Integer resultMsg)	DOM 2
Methods	
void **addFLUTEListener**(String multicastAddress)	
void **addFLUTEListenerTags**(String multicastAddress, String tags, String downloadCallBack)	
StringCollection **getFLUTEListeners**()	
String **getTags**(String multicastAddress)	
void **removeFLUTEListener**(String multicastAddress)	

15.2 Miscellaneous API: StatusView object

Methods
Integer **getMinimumItemWidth**(String state)
Integer **getMinimumItemHeight**(String state)

15.3 Miscellaneous API: Capabilities object

Properties
readonly Document **xmlCapabilities**
readonly Number **extraSDVideoDecodes**
readonly Number **extraHDVideoDecodes**
Methods
Boolean **hasCapability**(String profileName)

15.4 Miscellaneous API: Navigator

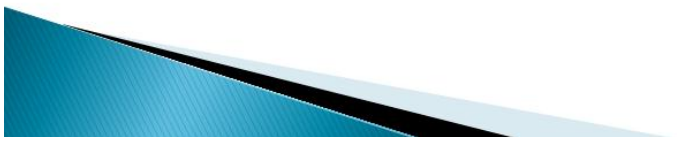
Properties
readonly String **appName**
readonly String **appVersion**

15.5 Miscellaneous API: Debug Print API

Methods
void **debug**(DOMString arg)

16.1 Shared Utilities: StringCollection

Properties
readonly Integer **length**
Methods
String **item**(Integer index)



16.2 Shared Utilities: Programme

Constants
ID_TVA_CRID
ID_DVB_EVENT
Properties
String **name**
String **longName**
String **description**
String **longDescription**
Integer **startTime**
Integer **duration**
String **channelID**
Integer **episode**
Integer **totalEpisodes**
String **programmeID**
Integer **programmeIDType**
readonly ParentalRatingCollection **parentalRatings**

16.2 Shared Utilities: Programme

Properties (extension)
readonly Channel **channel**
readonly Boolean **blocked**
readonly Integer **showType**
readonly Boolean **subtitles**
readonly Boolean **isHD**
readonly Integer **audioType**
readonly Boolean **isMultilingual**
readonly StringCollection **genre**
readonly Boolean **hasRecording**
readonly StringCollection **audioLanguages**
readonly StringCollection **subtitleLanguages**
readonly Boolean **locked**
readonly ScheduledRecording **recording**
Methods (extension)
String **getField**(String fieldId)
StringCollection **getSIDescriptors**(Integer descriptorTag, Integer descriptorTagExtension)

16.3 Shared Utilities: ProgrammeCollection

Properties
readonly Integer **length**
Methods
Programme **item**(Integer index)

16.4 Shared Utilities: DiscInfo

Properties
readonly Integer **free**
readonly Integer **total**
readonly Integer **reserved**

16.5 Shared Utilities: Media Playback extensions

Constants	
COMPONENT_TYPE_VIDEO	
COMPONENT_TYPE_AUDIO	
COMPONENT_TYPE_SUBTITLE	
Properties	
function **onSelectedComponentChanged**(Integer componentType)	DOM 2
Methods	
AVComponentCollection **getComponents**(Integer componentType)	
AVComponentCollection **getCurrentActiveComponents**(Integer componentType)	
void **selectComponent**(AVComponent component)	
void **unselectComponent**(AVComponent component)	
void **selectComponent**(Integer componentType)	
void **unselectComponent**(Integer componentType)	

ANEXO III) PRÁCTICA 6 TECNOLOGÍAS E INSTALACIONES DE VÍDEO "HBBTV: INTERACTIVIDAD EN TELEVISIÓN"

PRÁCTICAS DE TECNOLOGÍAS E INSTALACIONES DE VÍDEO

6º Semestre de Grado en Ingeniería en Tecnologías de Telecomunicación

HBBTV: INTERACTIVIDAD EN TELEVISIÓN

Curso 2012/2013

Mikel Sagues García

Imanol Eslava Arce

Contenido

1. INTRODUCCIÓN

El objetivo de la presente práctica es realizar una sencilla aplicación HbbTV [1]. Para ello, se dispone de un ejemplo ampliamente documentado, el cual debe ser modificado para incorporar nuevas funcionalidades. Además, los vídeos realizados a lo largo de la asignatura deben poder ser visualizados a través de dicha aplicación HbbTV.

La aplicación se encuentra alojada en un servidor web. Concretamente, en la dirección:

http://130.206.170.120/TIV/TIV2013/EjemploAppHbbTV/index.php

La aplicación creada a partir del ejemplo proporcionado, debe alojarse en un directorio dentro de la misma raíz:

http://130.206.170.120/TIV/TIV2013/

Para ello, se ha habilitado el acceso a la carpeta "TIV2013" mediante un servidor ftp con los siguientes datos:

ftp://130.206.170.120

Usuario: tiv2013

Contraseña: tiv2013

De este modo, es posible acceder a dicha carpeta para subir la aplicación HbbTV mediante cualquier cliente ftp (Filezilla, Windows Explorer…).

A lo largo de este documento, se describe en detalle una sencilla aplicación HbbTV. Esta aplicación estará formada por una página principal, que constará de un menú (en este caso tendrá una única opción), un título y un pequeño cuadro que contendrá una imagen y un texto:

Al pulsar sobre la opción del menú, en este caso Hostal Loizu, saltaremos a una página diferente. En este caso, la página estará formada por un menú, en el que tendremos las opciones de iniciar un vídeo, de pararlo, de continuar reproduciéndolo y una última opción que nos permita volver a la página principal.

Al pulsar sobre iniciar el vídeo, este comenzará a reproducirse en la parte derecha de la página.

Además, esta página también tendrá un cuadro de texto con una breve descripción de algo relacionada con el vídeo. En este caso como el vídeo es del Hostal Loizu, en el cuadro de texto aparece una descripción de dicho hotel.

Se propone la visualización de la aplicación ejemplo en el emulador FireHbbTV [2] (Addon de Mozilla Firefox) y en el emulador HbbTV de Opera [3]. En el **Anexo I,** se dispone de una introducción a dichos emuladores HbbTV.

Una vez hemos conocido como es nuestra primera aplicación HbbTV, vamos a tratar de explicar cómo ha sido implementada.

Lo primero que hay que tener en cuenta a la hora de realizar una primera aplicación HbbTV es los lenguajes de programación sobre los que se asienta dicho estándar. En un principio, para programar HbbTV basta con conocer JavaScript [4] y HTML [5]. Sin embargo, para hacer una aplicación dinámica, vamos a tener que tener nociones de PHP [6] además de algo de CSS [7].

Por este motivo, la base de nuestra aplicación va a estar formada por varios ficheros, unos con extensión .php otros con extensión .css y otros con extensión .js . A estos archivos se les unen otros, que son librerías predefinidas en HbbTV. En nuestro caso, utilizamos la librería de *keycodes*, que es la que relaciona nuestro teclado del ordenador con el mando de control remoto. Lo que hace es una equivalencia entre algunas teclas del teclado y teclas del mando a distancia. Por ejemplo, podemos asignar la tecla *enter* del ordenador con la tecla *OK* de nuestro control remoto.

Los archivos principales que conforman esta aplicación son:

- base.css
- base.js
- base.php
- index.php
- keycodes.js
- LOIZU/index.php

A continuación, se detalla el contenido de cada uno de estos ficheros que componen la aplicación.

2. BASE CSS

Fichero: base.css

Las **hojas de estilo en cascada** hacen referencia a un lenguaje de hojas de estilos usado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas. Su aplicación más común es dar estilo a páginas web escritas en lenguaje HTML y XHTML.

En este caso nos hemos limitado a hacer algo lo más sencillo posible. Lo primero que hemos hecho ha sido definir el tamaño y color del fondo de la base (luego en cada página de nuestra aplicación podremos cambiar el fondo de la página).

```
body {
    background-color: #d83101;
    height: 1280px;
    margin: 0px;
    padding: 0px;
    width: 720px;
}
```

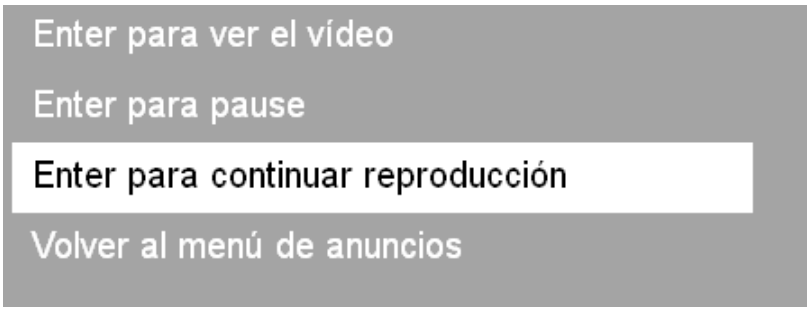
A continuación cuadros de texto y de imagen que luego podremos utilizar en nuestra aplicación.

```
div {
    overflow: hidden;
    position: absolute;
    font: 20px sans-serif;
    margin: 0;
    padding: 0;
    color: #ff0000;
}
.txtlg {
    font-size: 22px;
    font-weight: bold;
}
img {
    border: 0;
    margin: 0;
    padding: 0;
}
div.txtdiv {
    line-height: 26px;
    color: #ffffff;
}
div.imgdiv {
    background-repeat: no-repeat;
}
```

Y por último hemos definido un simple menú, que será el que nos permita posteriormente hacer saltos a diferentes páginas en función de la elección que hagamos sobre él.

```
ul.menu {
    padding: 0;
    margin: 0;
    position: absolute;
    width: 400px;
}
.menu li {
    display: none;
    overflow: hidden;
    width: 400px;
    height: 34px;
    text-align: left;
    padding-left: 11px;
    padding-top: 5px;
    font: 20px sans-serif;
    color: #ffffff;
    background-color: #A4A4A4;
}
.menu .lisel {
    color: #000000;
    background-color: #ffffff;
}
```

Lo más destacable de este menú tan sencillo es que lo hemos preparado de forma que la opción del menú sobre la que nos encontremos este seleccionada. Es decir, la opción en la que estemos tendrá un fondo diferente del resto, y así de esta forma sabremos en qué punto del menú nos encontramos.



3. BASE PHP

Fichero: base.php

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. En nuestro caso, nuestro archivo base.php va a constar de cuatro funciones.

La primera de ellas funciones va a ser ***sendContentType().*** Con esta primera función lo que hacemos es definir el tipo de contenido que va a tener nuestra aplicación HbbTV.

```
function sendContentType() {
   header('Pragma: no-cache');
   header('Cache-Control: no-cache');
   header('Content-Style-Type: text/css');
   $uagent = strtolower($_SERVER['HTTP_USER_AGENT']);
   if (strstr($uagent, 'firefox') || strstr($uagent, 'chrome')) {
      header('Content-Type: application/xhtml+xml; charset=UTF-8');
   } else {
      header('Content-Type: application/vnd.hbbtv.xhtml+xml; charset=UTF-8');
   }
}
```

La función ***openDocument()*** tiene como objetivo definir las especificaciones que se van a utilizar en nuestra la aplicación HbbTV. Esta segunda función lo que hace es definir la versión HbbTV, la API utilizada, y la localización de las diferentes ficheros de configuración utilizados en la aplicación.

```
function openDocument($title=TITLE, $allscripts=1, $addheaders='') {
  global $ROOTDIR;
  echo '<?xml version="1.0" encoding="utf-8" ?>'."\n";
  echo '<!DOCTYPE html PUBLIC "-//HbbTV//1.1.1//EN" "http://www.hbbtv.org/dtd/HbbTV-1.1.1.dtd">'."\n";
  echo "<html xmlns=\"http://www.w3.org/1999/xhtml\" xml:lang=\"en\" lang=\"en\">\n";
  echo "<head>\n";
  echo "<title>$title</title>\n".$addheaders;
  echo "<meta http-equiv=\"content-type\" content=\"Content-Type: application/vnd.hbbtv.xhtml+xml; charset=UTF-8\" />\n";
  echo "<link rel=\"stylesheet\" type=\"text/css\" href=\"$ROOTDIR/base.css\" />\n";
  echo '<script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>';
  if ($allscripts) {
    echo "<script type=\"text/javascript\" src=\"$ROOTDIR/keycodes.js\"></script>\n";
    echo "<script type=\"text/javascript\" src=\"$ROOTDIR/base.js\"></script>\n";
  }
```

La tercera función implementada en base.php es ***apppmgrObject()***. Con esta función lo que hacemos es definir cómo es la gestión de nuestra aplicación: memoria, estado, permisos, etc. Y su vez también definimos la interfaz de usuario de configuración y la información de configuración.

```
function appmgrObject() {
  if ($_REQUEST['demo']) return '';
  return '<object id="appmgr" type="application/oipfApplicationManager" style="position: absolute;
}
```

La última función que definimos es la función ***videoObject()***. Con esta función lo que hacemos es definir un objeto de vídeo que posteriormente vamos a utilizar en nuestra aplicación.

```
function videoObject($left=0, $top=0, $width=1280, $height=720) {
  if ($_REQUEST['demo']) {
    global $ROOTDIR;
    return '<img id="video" style="position: absolute; left: '.$left.'px; top: '.$top.'px; width: '.$width.'px; height: '.$height.'px;"
  }
  return '<object id="video" type="video/broadcast" style="position: absolute; left: '.$left.'px; top: '.$top.'px; width: '.$width.'px;
```

4. BASE JAVASCRIPT

Fichero: base.js

JavaScript es un lenguaje de programación interpretado orientado a objetos, basado en prototipos, imperativo y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas

En nuestro caso, en este archivo base de JavaScript, vamos a definir funciones globales que van a poder ser utilizadas en todas las páginas que tenga nuestra aplicación. No serán las únicas, ya que en cada página de la aplicación se podrán crear nuevas funciones que serán utilizadas solo en esa página.

En este caso, nuestra base.js consta de 6 funciones, que vamos a ir desgranando a continuación:

La primera de nuestras funciones es ***initVideo().***

```
function initVideo() {
  try {
    document.getElementById('video').bindToCurrentChannel();
  } catch (e) {
    // ignore
  }
  try {
    document.getElementById('video').setFullScreen(false);
  } catch (e) {
    // ignore
  }
}
```

El método **getElementById** nos devuelve una referencia del objeto HTML que le pasamos como parámetro. En este caso el parámetro viene con la "id" video. La propiedad ***id*** es un identificador único para cualquier marca HTML que luego nos permite desde Javascript acceder a dicho elemento. Es decir, luego con HTML tendremos que definir la ***id*** video. A partir de este objeto accedemos al método ***bindToCurrentChannel()*** con el cual accedemos al canal actual. Lo siguiente que definimos es que la propiedad ***setFullScreen*** sea *falso*, porque no queremos que el vídeo aparezca en pantalla completa.

La siguiente función que encontramos es ***initApp().***

```
function initApp() {
  try {
    var app = document.getElementById('appmgr').getOwnerApplication(document);
    app.show();
    app.activate();
  } catch (e) {
    // ignore
  }
  setKeyset(0x1+0x2+0x4+0x8+0x10);
}
```

Con esta función lo que hacemos es iniciar la aplicación. Cuando llamemos a esta función activaremos nuestra aplicación. Lo primero que hacemos es obtener el permiso del "dueño" de la aplicación. Para ello, definimos la variable ***app*** y la igualamos con la obtención del permiso.

A continuación le definimos la propiedad ***show*** para que sea visible y propiedad ***activate*** para iniciar la aplicación. En la última línea aparece una llamada a la función ***setKeyset*** con su parámetro de entrada que es una cierta máscara. Esta función la definiremos a continuación.

Nuestra tercera función es ***setKeyset().***

```
function setKeyset(mask) {
  // for HbbTV 0.5:
  try {
    var elemcfg = document.getElementById('oipfcfg');
    elemcfg.keyset.value = mask;
  } catch (e) {
    // ignore
  }
  try {
    var elemcfg = document.getElementById('oipfcfg');
    elemcfg.keyset.setValue(mask);
  } catch (e) {
    // ignore
  }
  // for HbbTV 1.0:
  try {
    var app = document.getElementById('appmgr').getOwnerApplication(document);
    app.privateData.keyset.setValue(mask);
    app.privateData.keyset.value = mask;
  } catch (e) {
    // ignore
  }
}
```

Con esta función lo que hacemos es igualar una variable a la configuración que hemos definido para el objeto de nuestra aplicación. Luego definimos esa variable como nuestra máscara. En este caso, hay que hacer dos variantes pues en función de la versión que utilicemos de HbbTV hay que escribir un código u otro.

A continuación aparece la función ***registerKeyEventListener().***

```
function registerKeyEventListener() {
  document.addEventListener("keydown", function(e) {
    if (handleKeyCode(e.keyCode)) {
      e.preventDefault();
    }
  }, false);
}
```

Con esta función lo que hacemos es detectar un evento del teclado registrando un listener [8]. Definimos la función que detectará el evento, definimos una función de prevención de errores y por último diremos que esta función está, por el momento, desactivada.

La siguiente función es ***menuInit().***

```
function menuInit() {
  opts = document.getElementById('menu').getElementsByTagName('li');
  menuSelect(0);
}
```

En esta función igualamos una variable, llamada ***opts***, a la selección que hagamos en el menú. Accedemos al menú mediante el método ***getElementById('menu')*** y a continuación a cada uno de los elementos del menú con la propiedad **getElementsByTagName('li').** En la siguiente línea hacemos una llamada a la función ***menuSelect()***, y le damos como parámetro de entrada el valor cero. Esta función será la última que se defina en el archivo base y será explicada a continuación:

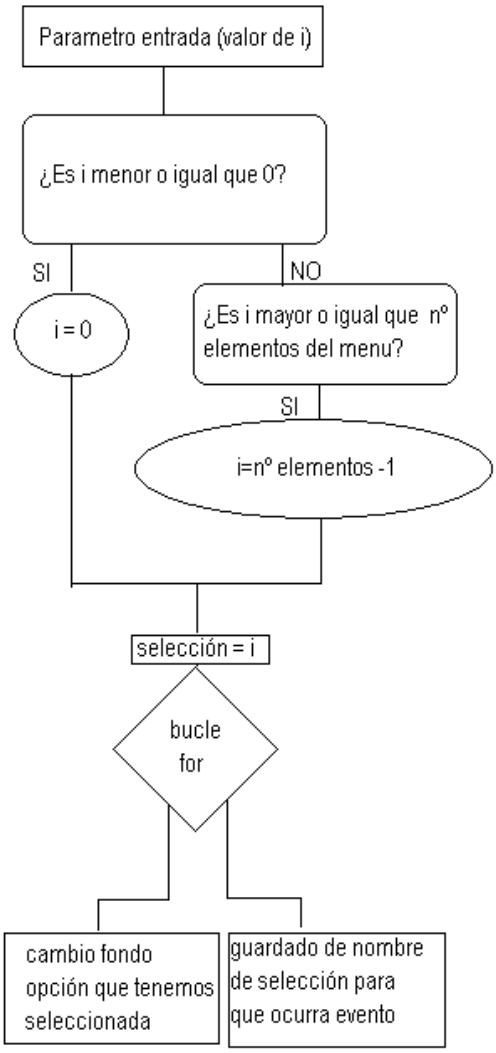
```
function menuSelect(i) {
  if (i<=0) {
    i = 0;
  } else if (i>=opts.length) {
    i = opts.length-1;
  }
  selected = i;
  var scroll = Math.max(0, Math.min(opts.length-3, selected-1));
  for (i=0; i<opts.length; i++) {
    opts[i].style.display = (i>=scroll && i<scroll+3) ? 'block' : 'none';
    opts[i].className = selected==i ? 'lisel' : '';
  }
}
```

Como vemos, en esta función tenemos como parámetro de entrada un valor entero. Supongamos que es el valor 0 como ocurre en este caso. Abrimos un condicional, en el cual si el valor es menor o igual que cero va a tomar el valor 0 (con esto accedemos el primer elemento del menú). Si es mayor o igual que cero entrará en otro condicional, y si el valor es mayor o igual que la longitud (número de elementos del menú), el valor de la variable es igual al número de elementos totales menos 1 (con esto accedemos al último elemento del menú). Restar en este caso es ir hacia abajo en el menú, por lo que si al total le restamos uno nos aseguramos de que estamos en el último.

A continuación definimos una variable ***scroll***. Esta variable será el valor mayor de la comparación entre 0 y el mínimo de número de elementos menos 3 y selección menos 1. Con

esta variable lo que vamos a hacer es que se vaya seleccionando la variable del menú sobre la que estamos en ese momento. Es una forma de saber qué elemento del menú tenemos seleccionado. La variable ***scroll+3*** indica el número de elementos del menú que son visibles. Por último, guardamos el nombre de la variable seleccionada para saber qué hemos seleccionado en el menú**,** guardando el nombre en id ***lisel***.

Este método es, por lo tanto, el que nos permite ver qué opción del menú está seleccionada, y a la vez es el que guarda el nombre de la opción seleccionada por el usuario, para de esta forma realizar la acción apropiada en función de la elección del usuario. Para clarificar todavía más todo lo anterior, en la siguiente figura se puede ver un pequeño diagrama del planteamiento realizado para crear la función ***menuSelect***.

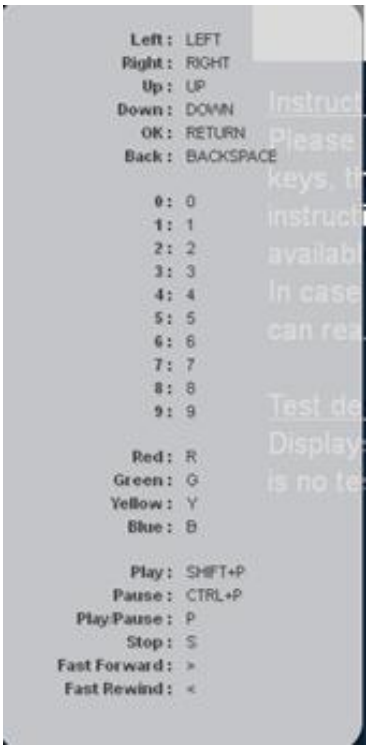


5. CONTROL DE EVENTOS DEL MANDO A DISTANCIA

Fichero: keycodes.js

Este archivo programado en JavaScript es una librería en la que se asocian los diferentes eventos del mando a distancia a variables que serán utilizadas por otras funciones como por ejemplo ***handleKeyCode()***.

En caso de que los eventos del mando a distancia no estén definidos, la función realiza una asociación entre las teclas del ordenador y los botones del control remoto. Su objetivo es definir la funcionalidad de las teclas del teclado en caso de que la aplicación se ejecute en un emulador HbbTV que no tenga definidas dichas asociaciones.



Como vemos, asocia las flechas de navegación del teclado con las del control remoto, el *enter* del teclado con el *OK* del mando, la barra de espacios del teclado del ordenador con la flecha de volver del mando, etc. Los números son iguales, y los botones de colores del mando los asocia con la inicial de cada color en inglés.

En la siguiente figura se puede ver una parte del código. En primer lugar se comprueba que los eventos del mando a distancia están definidos, en cuyo caso se asigna a cada uno una variable. En caso de no estar definidos, se asocian dichos valores a las teclas del teclado.

```
if (typeof(KeyEvent.VK_0)!='undefined') {
  var VK_0 = KeyEvent.VK_0;
  var VK_1 = KeyEvent.VK_1;
  var VK_2 = KeyEvent.VK_2;
  var VK_3 = KeyEvent.VK_3;
  var VK_4 = KeyEvent.VK_4;
  var VK_5 = KeyEvent.VK_5;
  var VK_6 = KeyEvent.VK_6;
  var VK_7 = KeyEvent.VK_7;
  var VK_8 = KeyEvent.VK_8;
  var VK_9 = KeyEvent.VK_9;
}
```

```
if (typeof(VK_0)=='undefined') {
  var VK_0 = 0x30;
  var VK_1 = 0x31;
  var VK_2 = 0x32;
  var VK_3 = 0x33;
  var VK_4 = 0x34;
  var VK_5 = 0x35;
  var VK_6 = 0x36;
  var VK_7 = 0x37;
  var VK_8 = 0x38;
  var VK_9 = 0x39;
}
```

6. PÁGINA PRINCIPAL DE LA APLICACIÓN

Fichero: index.php

Este archivo será la página principal de nuestra aplicación. Cuando se abra la aplicación, esto será lo que vea el usuario. Lo primero que hacemos es definir ciertas sentencias en **PHP**. En él definimos la raíz de este documento y abrimos el documento al que hace referencia esta raíz.

```php
<?php
$ROOTDIR='.';
require("$ROOTDIR/base.php");
$referer = $_SERVER['HTTP_REFERER'];
$i = strrpos($referer, '/');
$referer = substr($referer, 0, $i);
$referer = substr(strrchr($referer, '/'), 1);
$referer = addcslashes($referer, "\0..\37'\\");

sendContentType();
openDocument();
?>
```

A continuación, aparece el código **JavaScript**. Aquí lo primero que hacemos es llamar a las funciones del archivo base que vamos a utilizar en la página.

```javascript
window.onload = function() {
  menuInit();
  registerKeyEventListener();
  setDescr();
  initApp();
  nameselect('<?php echo $referer; ?>');
```

En primer lugar definimos la funcionalidad de la ***window.onload()***, de tal forma que al cargarse la aplicación se ejecuten ciertas funciones. Es decir, se definen las funciones que se van a ejecutar cuando se haya cargado la página completamente. En este caso cargamos el menú, llamamos a la función que registra los eventos de teclado e iniciamos la aplicación. Además, aparecen dos funciones que no habíamos definido, la primera de ellas es ***setDescr().***

```
function setDescr() {
  document.getElementById('descr').innerHTML = opts[selected].getAttribute('descr');
}
```

Con esta función lo que hacemos es guardar un atributo de la opción del menú para así poder crear un enlace entre la selección que hacemos en el menú y la página (debe llevar su mismo nombre) a la que vamos a acceder al elegir esa opción del menú.

La siguiente función que aparece es ***nameselect:***

```
function nameselect(snam) {
  if (!snam) return;
  for (var i=0; i<opts.length; i++) {
    var check = opts[i].getAttribute('name');
    if (check==snam) {
      menuSelect(i);
      setDescr();
      break;
    }
  }
}
```

Con esta función nos quedaremos definitivamente con una selección. Será la que guarde definitivamente el nombre de la selección y la que enlace con la página que lleva su nombre. Por eso, en ella hay una llamada tanto a la función ***menuSelect()*** como a la función ***SetDescr().***

La función ***handleKeyCode()*** se emplea para navegar por el menú. En este caso como el menú solo tiene una opción no tiene mucho sentido, pero se añade a modo de ejemplo, para ilustrar la forma en la que se podría implementar esta funcionalidad.

```
function handleKeyCode(kc) {
  if (kc==VK_UP) {
    menuSelect(selected-1);
    setDescr();
    return true;
  } else if (kc==VK_DOWN) {
    menuSelect(selected+1);
    setDescr();
    return true;
  } else if (kc==VK_LEFT){
    menuSelect(selected-9);
    setDescr();
    return true;
  } else if (kc==VK_RIGHT){
    menuSelect(selected+9);
    setDescr();
    return true;

  } else if (kc==VK_ENTER) {
    var liid = opts[selected].getAttribute('name');
    if (liid=='exit') {
      closeApp();
```

Con esta función lo que hacemos es un condicional. La primera parte del condicional dice que si se pulsa la tecla UP la selección suba uno, la segunda que si pulsas la tecla DOWN la selección del menú baje una posición, la tercera que si pulsas le tecla LEFT se suba 9 posiciones en el menú y la cuarta que si pulsas la tecla RIGHT se baje 9 posiciones en el menú. En la última condición lo que se hace es definir que si se pulsa la tecla ENTER, en la variable ***liid*** se guarde el nombre de la selección que se ha hecho (lo que utilizamos en otras funciones para el cambio a otra página). En este caso lo único que se hace es que al pulsar el ENTER se guarde el nombre de la selección para así saltar a otra página.

Se ha colocado una condición que dice que si la variable ***liid*** definida es igual a exit se llame a una función ***closeApp()*** mediante la cual se cerrará la aplicación. En este caso, se ha implementado una **función *closeApp()*** que cierra la aplicación y devuelve un mensaje de alerta diciendo *Cannot destroy application* en caso de que la aplicación no se cierre con éxito:

```
function closeApp() {
  try {
    var app = document.getElementById('appmgr').getOwnerApplication(document);
    app.destroyApplication();
    return;
  } catch (e) {
    alert('Cannot destroy application');
  }
}
```

Seguido del código en JavaScript aparece una sentencia en PHP que utilizamos para llamar a la función ***appmgrObject()*** mediante la cual se crea un objeto de tipo objeto *application manager* definido en el estándar OIPF [9]:

```
<?php
echo appmgrObject();
?>
```

A continuación aparece el código **HTML.** Lo primero que hemos hecho ha sido definir el color del fondo de nuestra página:

```
<div style="left: 0px; top: 0px; width: 1280px; height: 720px; background-color: #A4A4A4;" />
```

A continuación hemos definido un pequeño cuadro de texto y lo hemos colocado en la parte superior izquierda, a modo de título:

```
<div class="txtdiv txtlg" style="left: 111px; top: 60px; width: 500px; height: 30px;">ANUNCIOS</div>
```

Lo siguiente que hemos hecho ha sido definir un espacio, en la parte superior derecha, en el que hemos introducido un cuadro de texto y otro de imagen.

```
<div style="left: 690px; top: 100px; width: 450px; height: 300px; background-color: #A4A4A4;">
  <div class="txtdiv" style="left: 10px; top: 4px; width: 400px; height: 30px; color: #000000;">Producción Multimedia Interactiva:</div>
  <div class="imgdiv" style="left: 60px; top: 54px; width:224px; height: 200px; background-image: url(Logo_UPNA.svg.png);"></div>
</div>
```

Obteniendo el siguiente resultado:

Por último hemos definido el menú de la siguiente forma:

```
<ul id="menu" class="menu" style="left: 100px; top: 100px;">
  <li name="LOIZU" >Hostal Loizu</li>
</ul>
```

Si quisiéramos que nuestro menú tuviera más opciones bastaría con añadir opciones debajo de la sentencia ***name=Loizu***, escribiéndolas de la misma forma. Aquí un ejemplo de un menú con más opciones y el resultado:

```
<ul id="menu" class="menu" style="left: 100px; top: 100px;">
  <li name="Juan/index.php" >Juan</li>
  <li name="Pedro/index.php" >Pedro</li>
  <li name="María/index.php" >María</li>
</ul>
```

Como vemos, definimos ***name*** y le damos diferentes nombres. Como se ve, la variable ***name*** se iguala con la ruta que va a seguir la aplicación al pulsar en cada una de las opciones que tiene el menú.

7. PÁGINA SECUNDARIA

Fichero: LOIZU/index.php

Esta página no varía mucho de la principal creada anteriormente. La apariencia es básicamente la misma. Se ha añadido pequeño menú con la opción de empezar a reproducir un vídeo, poder pararlo, continuar con la reproducción y volver a la página principal. Por lo tanto el código va a ser similar con pequeñas variantes. Trataremos de explicar dichas variantes.

```
<?php
$ROOTDIR='..';
require("$ROOTDIR/base.php");
sendContentType();
openDocument();

?>
<script type="text/javascript">

window.onload = function() {
  menuInit();
  initVideo();
  registerKeyEventListener();
  initApp();
```

Como se ve, el código inicial es igual, con la única variación de que en este caso sí llamamos a la función ***initVideo()***. Además, se ha redefinido la función ***handleKeyCode(kc)*** de tal forma que, entre otras cosas, se ha implementado una opción de retorno a la página principal.

```
if (liid=='exit') {
  document.location.href = '../index.php';
```

Lo fundamental en esta página es la aparición del vídeo. La función que controla el vídeo es ***runStep(),*** la cual es llamada desde ***handleKeyCode(kc)*** en función de la selección realizada por el usuario. Para ello, se actúa sobre el objeto **video** el cual es almacenado en la variable ***vid***. La propiedad ***data*** será la que igualemos con la dirección en la que está nuestro vídeo. Más adelante se explicará el formato en el que este vídeo puede ser codificado.

La función ***play***() es la que controla la reproducción del vídeo, recibiendo un parámetro a través del cual se ajusta la velocidad de reproducción. Si pasamos un valor 0 el vídeo se detiene (*pause*) y si pasamos un valor de 1 el vídeo se reproduce a velocidad normal. Para realizar esta función se ha definido una función auxiliar ***setSpeed()*.**

Por último ponemos la variable ***playing*** a ***true***, pues antes la habíamos definido como falsa:

```
var playing = false;
```

A continuación, vemos el código tanto de la función ***runStep()*** como de la función ***setSpeed()*.**

```
function runStep(name) {
  if (name=='start') {
    try {
      var vid = document.getElementById('video');
      vid.stop();
      vid.data = 'http://130.206.170.120/TIV/ImanolEslava/PMMI%20APP/LOIZU/si.mp4';
      vid.play(1);
      playing = true;
    } catch (e) {

    }
  } else if (name=='pause') {
    setSpeed(0);
  } else if (name=='play') {
    setSpeed(1);
  }
}

function setSpeed(fact) {

    var vid = document.getElementById('video');
    vid.play(fact);
}
```

Para poder visualizar el vídeo, debemos definir un objeto apropiado y posicionarlo en la pantalla:

```
<object id="video" type="video/mp4" style="position: absolute; left: 600px; top: 120px; width: 400px; height: 400px;"></object>
```

El resto de código HTML es muy similar, habiéndose añadido una pequeña explicación sobre el vídeo, el formato y su codificación.

8. FORMATOS DE VÍDEO EN HBBTV

Los formatos de vídeo soportados por el estándar HbbTV están especificados por la especificación OIPF. Se pueden encontrar en la sección 5 de este documento. Vemos una tabla de formatos de vídeo y audio para 25 Hz

System Format	Video Format	Audio Format	Mime Type
TS	AVC_HD_25 AVC_SD_25 AVC_SP_25	HEAAC HEAAC2 HEAAC_MPS MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3 DTS	video/mpeg or video/mp2t
TTS	AVC_HD_25 AVC_SD_25 AVC_SP_25	HEAAC HEAAC2 HEAAC_MPS MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3 DTS	video/vnd.dlna.mpeg-tts
MP4	AVC_HD_25 AVC_SD_25 AVC_SP_25	HEAAC HEAAC2 HEAAC_MPS MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3 DTS	video/mp4
TS	MPEG2_SD_25 MPEG2_SP_25	MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3	video/mpeg or video/mp2t
TTS	MPEG2_SD_25 MPEG2_SP_25	MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3	video/vnd.dlna.mpeg-tts

Table 1: A/V Media Formats for 25Hz video system

Aquí la misma tabla para formato de 30 Hz:

System Format	Video Format	Audio Format	Mime Type
TS	AVC_HD_30 AVC_SD_30 AVC_SP_30	HEAAC HEAAC2 HEAAC_MPS AC3 E-AC3 DTS	video/mpeg or video/mp2t
TTS	AVC_HD_30 AVC_SD_30 AVC_SP_30	HEAAC HEAAC2 HEAAC_MPS AC3 E-AC3 DTS	video/vnd.dlna.mpeg-tts
MP4	AVC_HD_30 AVC_SD_30 AVC_SP_30	HEAAC HEAAC2 HEAAC_MPS AC3 E-AC3 DTS	video/mp4

Table 2: A/V Media Formats for 30Hz video system

El formato MP4 está definido en la Parte 14 de MPEG-4 [10]. Es un formato estándar de contenedor multimedia especificado como parte del estándar IEC. Se utiliza para almacenar los formatos audiovisuales especificados por ISO/IEC y el grupo MPEG (Moving Picture Experts Group) al igual que otros formatos audiovisuales. Se utiliza típicamente para almacenar datos en archivos para ordenadores, para transmitir flujos audiovisuales y en muchas otras aplicaciones.

Comúnmente se utiliza para encapsular contenido de audio y vídeo digital, pero también puede ser utilizado para combinar muchos más tipos de contenido multimedia, tales como audio múltiple, vídeos, subtítulos e imágenes fijas.

En nuestro caso hemos utilizado un contenedor MP4 que contenía vídeo codificado con H.264 (o lo que es lo mismo, MPEG-4 Parte 10 [11]), que sí está soportado por HbbTV. H.264 es una norma que define un códec de vídeo de alta compresión. La intención del proyecto H.264/AVC fue la de crear un estándar capaz de proporcionar una buena calidad de imagen con tasas binarias notablemente inferiores a los estándares previos (MPEG-2, H.263 o MPEG-4 parte 2), además de no incrementar la complejidad de su diseño.

Proyecto Final de Carrera
Hybrid Broadcast Broadband TV

Estudio del estándar de televisión digital interactiva HbbTV e implementación de aplicación final

eslava33@hotmail.com
Imanol Eslava Arce

Índice

- Introducción
  - Interactividad en televisión
  - MHP
  - Smart TV
- Estándar HbbTV
  - Introducción
  - Arquitectura
  - Especificaciones
  - Tipos de aplicaciones
  - Emuladores HbbTV

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

eslava33@hotmail.com
Imanol Eslava Arce

Índice

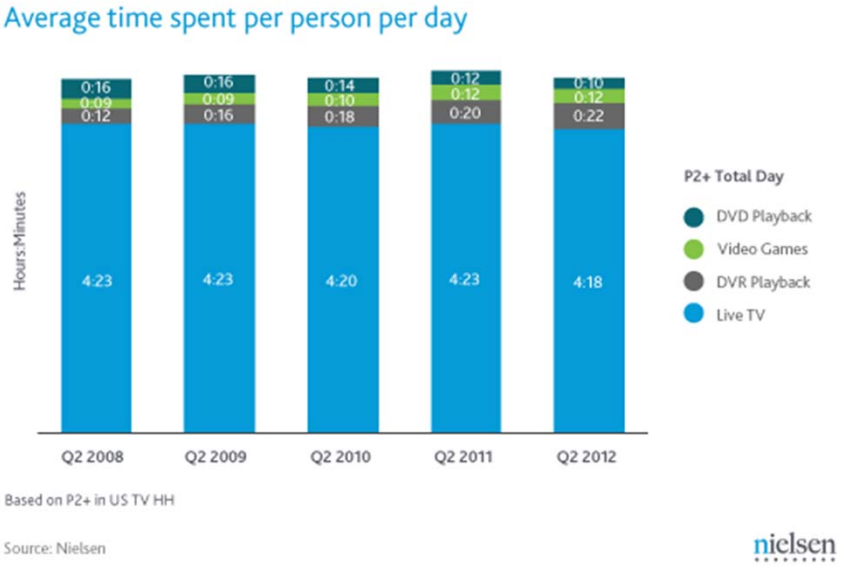
- Aprendiendo HbbTV
  - Familiarización con las tecnologías
  - Códigos ajenos y modificaciones
  - Primeras pruebas
  - Práctica tema 6 Tecnologías e Instalaciones Vídeo
- Aplicación final
  - Objetivo
  - Tecnologías
  - Demostración
- Conclusiones

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

eslava33@hotmail.com
Imanol Eslava Arce

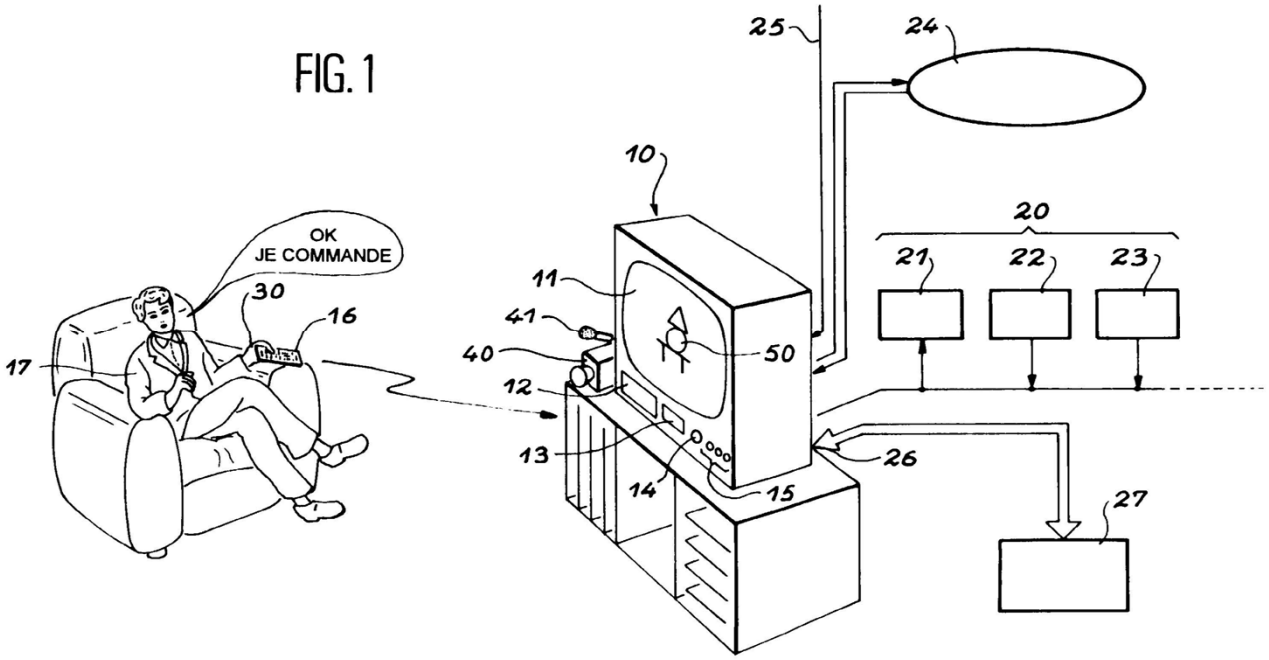
HbbTV
Interactividad en Televisión

- Problema: el PC se come paulatinamente el mercado de entretenimiento audiovisual
- Los fabricantes de televisores quieren seguir viviendo de fabricar televisores
- Solución: convergencia
- Aunar la experiencia multimedia
  - Contenido de broadcasting
  - Contenido en Internet
- ¿Dónde? ¡En el televisor!
- ¿Cómo?
  - Con el mando
  - Con una interfaz unificada
  - Convirtiendo la TV en un ordenador de propósito específico



Introducción

- **Frikidato: primera patente en Francia, 1994**
  - **Un sistema "inteligente" de televisión conectada con los sistemas de procesamiento de datos por medio de una red digital o analógica**

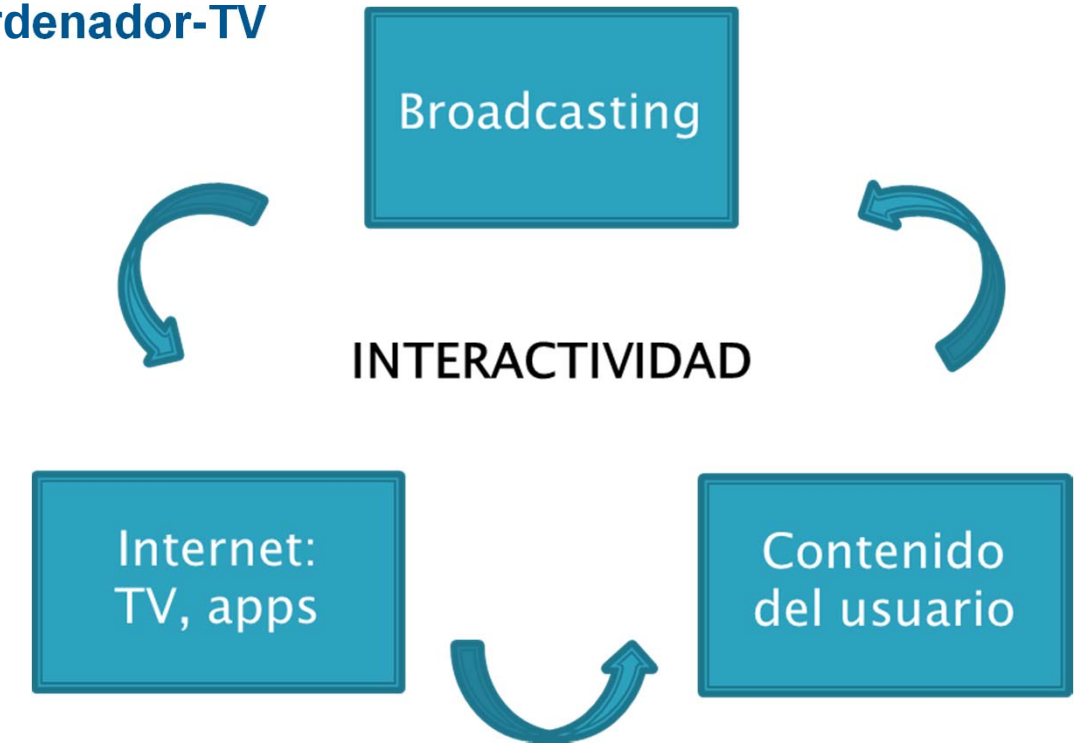


MHP: Multimedia Home Platform

- MHP está basada en JAVA
- Poco flexible y con capacidades limitadas
- ¡Se puede hablar de fracaso en toda regla!

Smart TV

- Integración de Internet en la TV
- Convergencia ordenador-TV
- Objetivos



¿Qué es HbbTV?

- ETSI TS 102 796, en junio de 2010.
- Hybrid Broadcast Broadband TV o HbbTV , es un proyecto paneuropeo de televisión híbrida cuyo objetivo es combinar las emisiones de televisión (broadcast) con servicios de banda ancha (broadband) para entregar al telespectador un servicio de entretenimiento a través de una pantalla de televisión.

- La Televisión Híbrida trata por tanto de proporcionar un servicio de televisión y de contenido Web mediante banda ancha.

- Abierta y neutral

¿Qué es HbbTV?

- **Nuevos servicios de entretenimiento:**
  - **Recuperación de programas de televisión u otros contenidos: Vídeo bajo demanda (VoD)**
  - **Publicidad interactiva**
  - **Información personalizada en el televisor**
  - **Votación**
  - **Juegos**
  - **Navegación Web**
  - **Redes sociales, etc.**
  - **Servicios relacionados con el propio programa:**
    - **Teletexto**
    - **Guía Electrónica de Programación (EPG)**

Smart TV vs HbbTV

– No compite con Smart TV: lo complementa

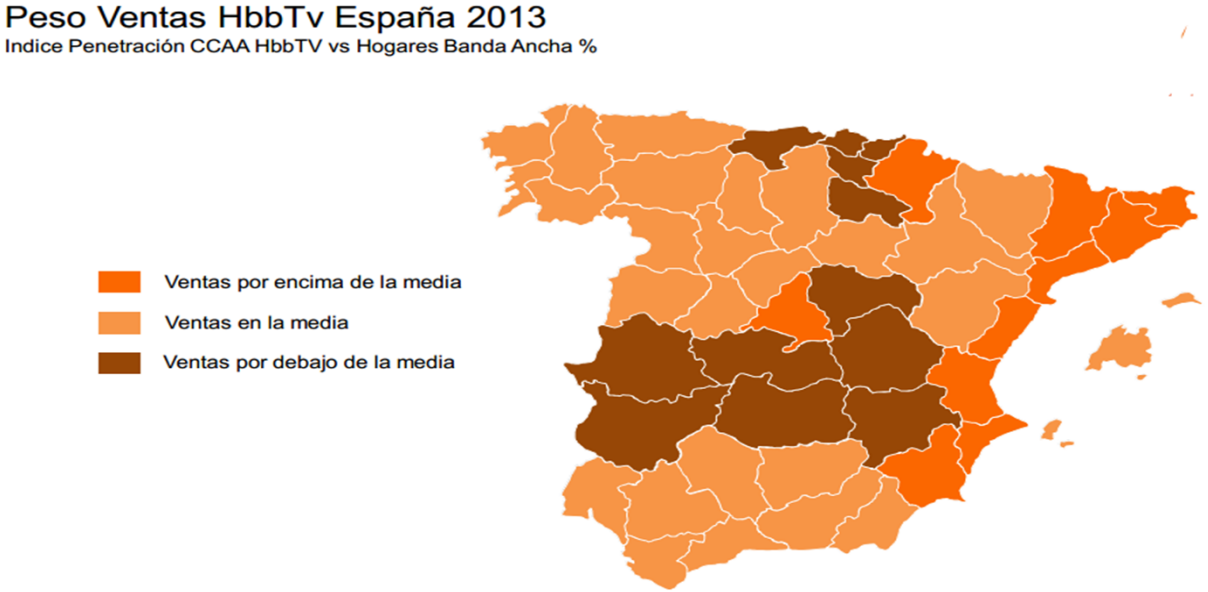
- Esencialmente, aporta al broadcaster facilidad para ofrecer sus contenidos:
  - Estándar abierto: diseño de la aplicación solo una vez!

– Middlewares de terceros frente a uno propio

- Ventajas para el fabricante
  - Desarrollo hardware únicamente
  - Productos más baratos
- Ventajas para el proveedor de contenidos
  - Una sola aplicación funciona en muchas marcas
- Desventajas para el fabricante
  - Imagen
  - Desarrollo más rígido
  - Reduce su capacidad de innovar

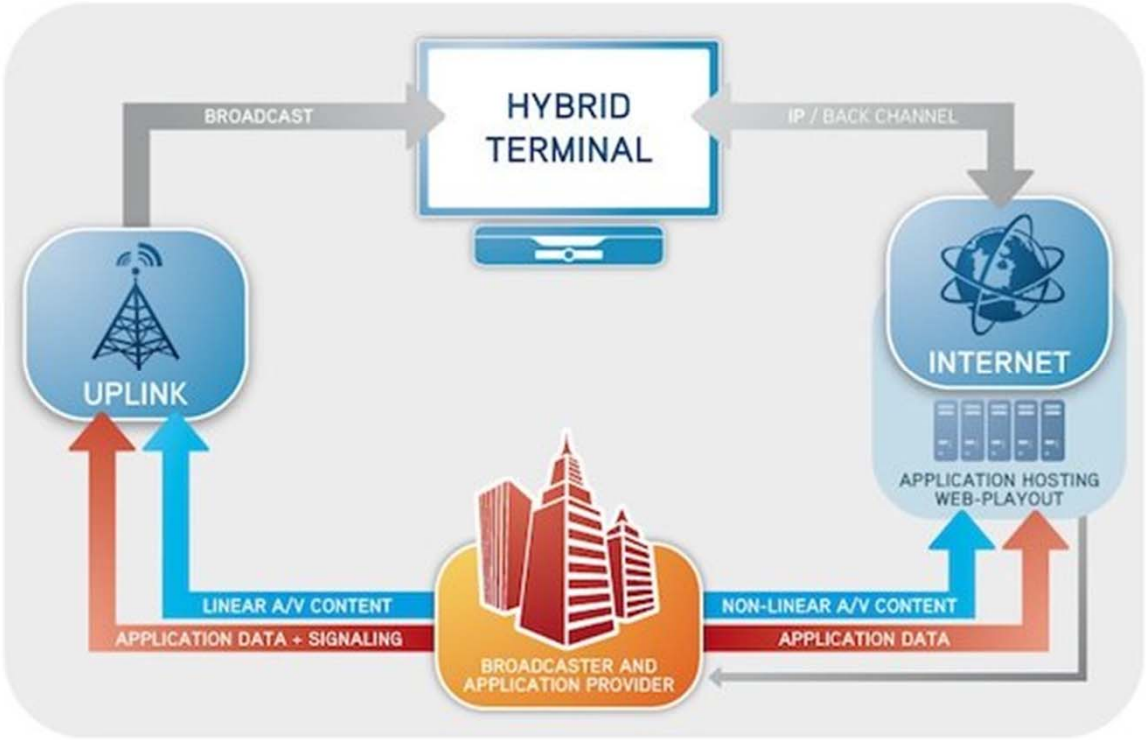
Estado actual en España

- En España hay 2,7 millones de 'smart TV' o televisores inteligentes, de los cuales aproximadamente 1,6 cuentan con el sistema HbbTV (más del 5% de los hogares españoles), con servicios horizontales de televisión conectada.
- Las comunidades en las que más televisores conectados se están vendiendo son Madrid, Cataluña, Comunidad Valenciana, Navarra y Murcia. A la cola encontraríamos Castilla-La Mancha, País Vasco, Rioja y Cantabria.



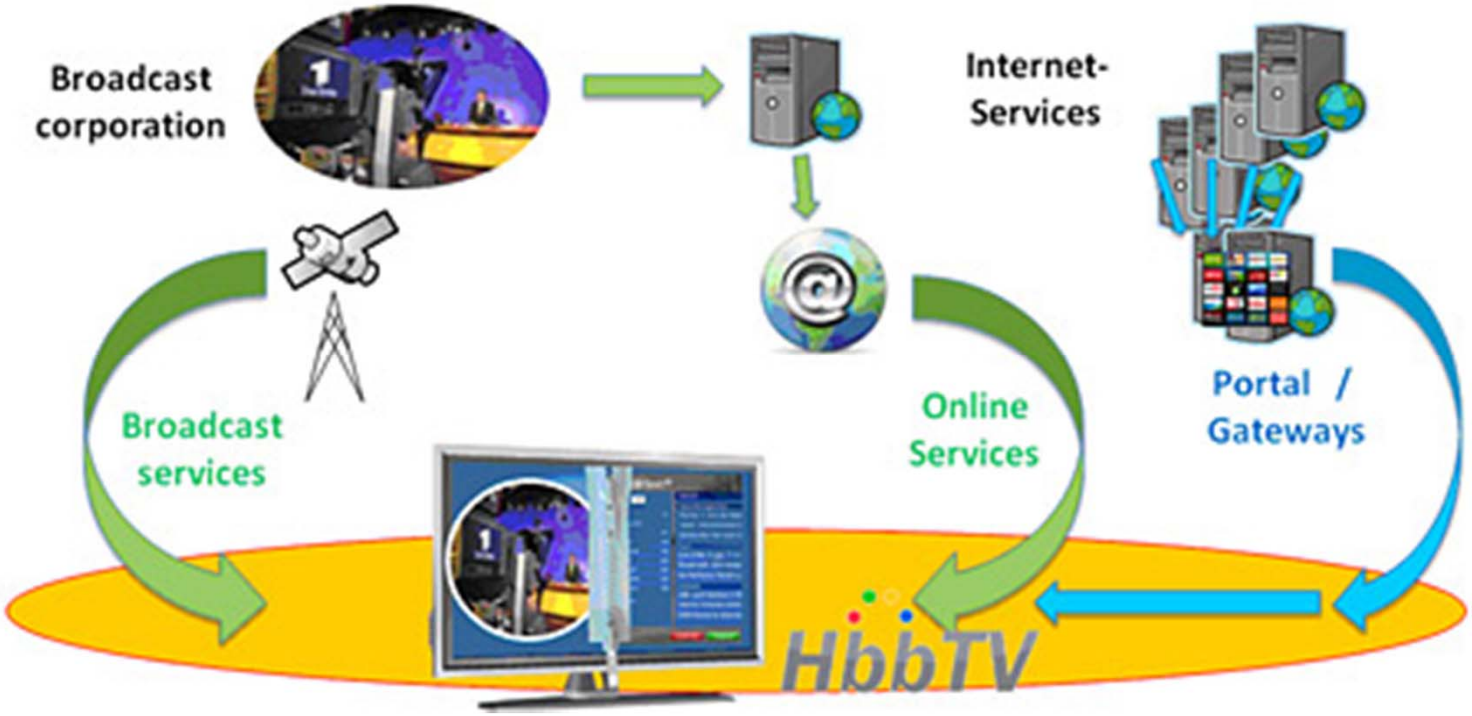
Arquitectura de una aplicación HbbTV

- Estándar abierto
- Convergencia: televisión + Internet



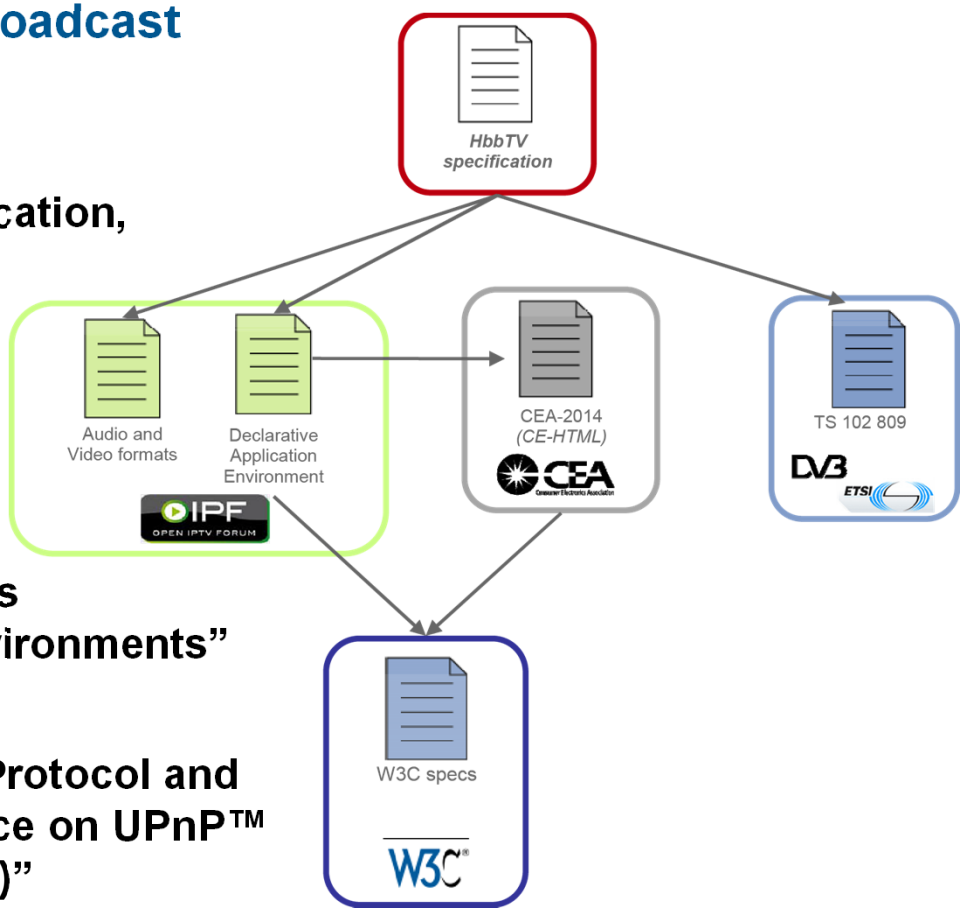
Arquitectura de una aplicación HbbTV

– Lo ideal sería que los fabricantes y proveedores permitieran un acceso libre a otros contenidos



Especificaciones de la especificación

- **ETSI TS 102 796 (V1.1.1): "Hybrid Broadcast Broadband TV"**
- **Normativo:**
  - **Open IPTV Forum Release 1 specification, volume 5 (V1.1): "Declarative Application Environment"**
  - **ETSI TS 102 809 (V1.1.1): "Digital Video Broadcasting (DVB); Signalling and carriage of interactive applications and services in Hybrid Broadcast/Broadband environments"**
- **Informativo:**
  - **CEA-2014 revision A: "Web-based Protocol and Framework for Remote User Interface on UPnP™ Networks and the Internet (Web4CE)"**



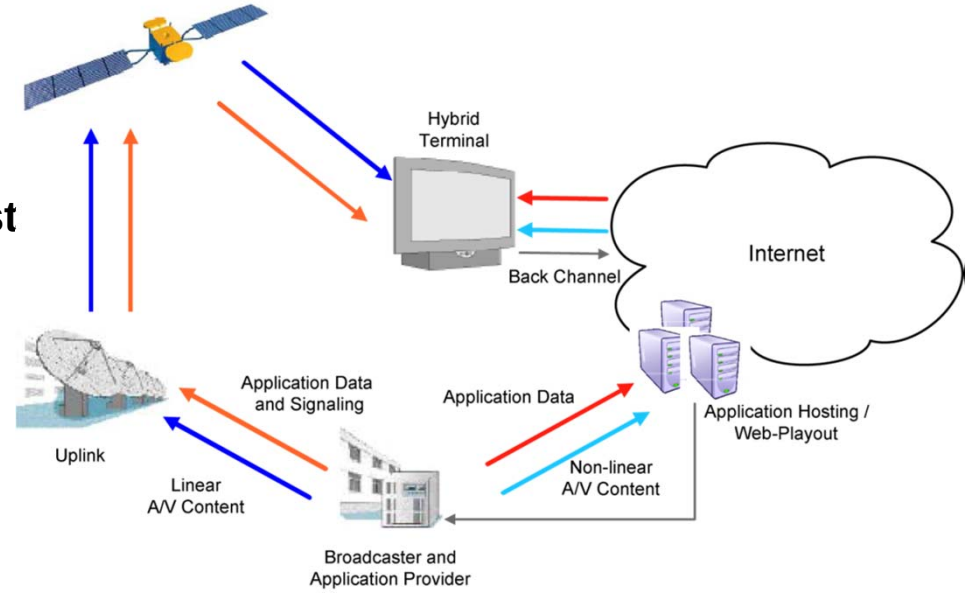
Especificaciones de la especificación

- **CEA-2014 + OIPF**
  - **Lenguajes:**
    - **XHTML**
    - **CSS**
    - **Javascript + AJAX**
  - **Video embedding: MPEG-TS, MP4**
  - **DOM event-handling**
  - **Formatos de imagen: JPEG, GIF, PNG**
  - **API de Javascript específica para TV**

Tipos de aplicación

- **Broadcast-related**
  - Banda ancha o canal de broadcast
  - Autostart o iniciada por el usuario
  - Conviven con el canal de broadcast asociado

- **Broadcast-independent**
  - Banda ancha
  - Se cierra el canal de broadcast



Emuladores

- **Opera HbbTV Emulator**
  - http://business.opera.com/solutions/tv/emulator
- **FireHbbTV**
  - https://addons.mozilla.org/es/firefox/addon/firehbbtv/
- **Ejemplos disponibles en la red**
  - MIT-xperts HBBTV testsuite
    - https://github.com/mitxp/HbbTV-Testsuite
    - http://itv.mit-xperts.com/hbbtvtest/
  - IRT
    - http://tv-html.irt.de
  - ARD
    - http://itv.ard.de/ardepg/index.php

¿Por dónde empezar?

- **Necesidad de conocer las tecnologías que aparecen en el estándar Hbbtv**
  - **Javascript**
  - **Html + Css**
  - **Php (no necesario pero importante)**

- **Conocer estas tecnologías va a permitir:**
  - **Estudiar código ajeno**
  - **Modificar código**
  - **Crear nuestros propios códigos**

Familiarización con las tecnologías

- **Desconocimiento**
  - Tutoriales desde cero
  - Proceso lento y laborioso, pero enriquecedor
- **Tutorial JavaScript**
  - Tutorial a través de una página web
  - Parte teórica de 100 temas y ejercicios resueltos y propuestos
  - Se completa con una serie de contenidos de vídeo
- **Tutorial Html+ Css**
  - Tutorial a través medio audivisual
  - 23 vídeos con explicaciones y ejercicios resueltos
  - Resolución de ejercicios propuestos
- **Tutorial Php**
  - No necesario pero fundamental para lograr objetivos
  - Tutorial a través medio audivisual
  - 21 vídeos con explicaciones y ejercicios resueltos
  - Resolución de ejercicios propuestos

Códigos ajenos y modificaciones

- **Mit-Xperts app**
- **Ofrece un tetsuite con diferentes funciones**
  - **Se pueden ver las opciones que se pueden incluir en una HbbTV APP**
  - **Se puede disponer del código para estudiar y ver como son implantadas dichas funciones**
  - **Utiliza librerías que son de gran utilidad para aplicaciones futuras propias**
- **Estudio exhaustivo del código para comprender archivos base, librerías, programas principales…**
- **Modificaciones en todos los archivos, tanto css, como php, como funciones javascript como en programas principales**

Primeras pruebas

- Se realiza un primera aplicación
- Pruebas muy básicas
  - Menú principal
  - Cuadros de texto e imagen
  - Profundidad de un salto (página principal y secundaria)
  - Pequeño juego
  - Menú envía a cada una de las páginas secundarias en función de la selección

Práctica tema 6 Tecnologías e Instalaciones Vídeo

- Aparece la posibilidad de que un tema de dicha asignatura sea HbbTV
- Clave : documentación de código!!
  - Documento para ingenieros en prácticas que desconocen la materia
  - Descripción punto por punto de cada una de las partes de código escrito para la creación de la aplicación

La primera de nuestras funciones es *function initVideo()*

```
function initVideo() {
  try {
    document.getElementById('video').bindToCurrentChannel();
  } catch (e) {
    // ignore
  }
  try {
    document.getElementById('video').setFullScreen(false);
  } catch (e) {
    // ignore
  }
}
```

El método **getElementById** nos retorna una referencia del objeto HTML que le pasamos como parámetro. En este caso el parámetro viene con la "id" video.

La **propiedad "id"** es un identificador único para cualquier marca HTML que luego nos permite desde Javascript acceder a dicho elemento. Es decir, luego con HTML tendremos que definir la "id" video.

A partir de este objeto accedemos a la propiedad *bindToCurrentChannel()* con la cual accedemos al canal actual.

Lo siguiente que definimos es que la propiedad *setFullScreen* sea falso, porque no queremos que el vídeo aparezca en pantalla completa.

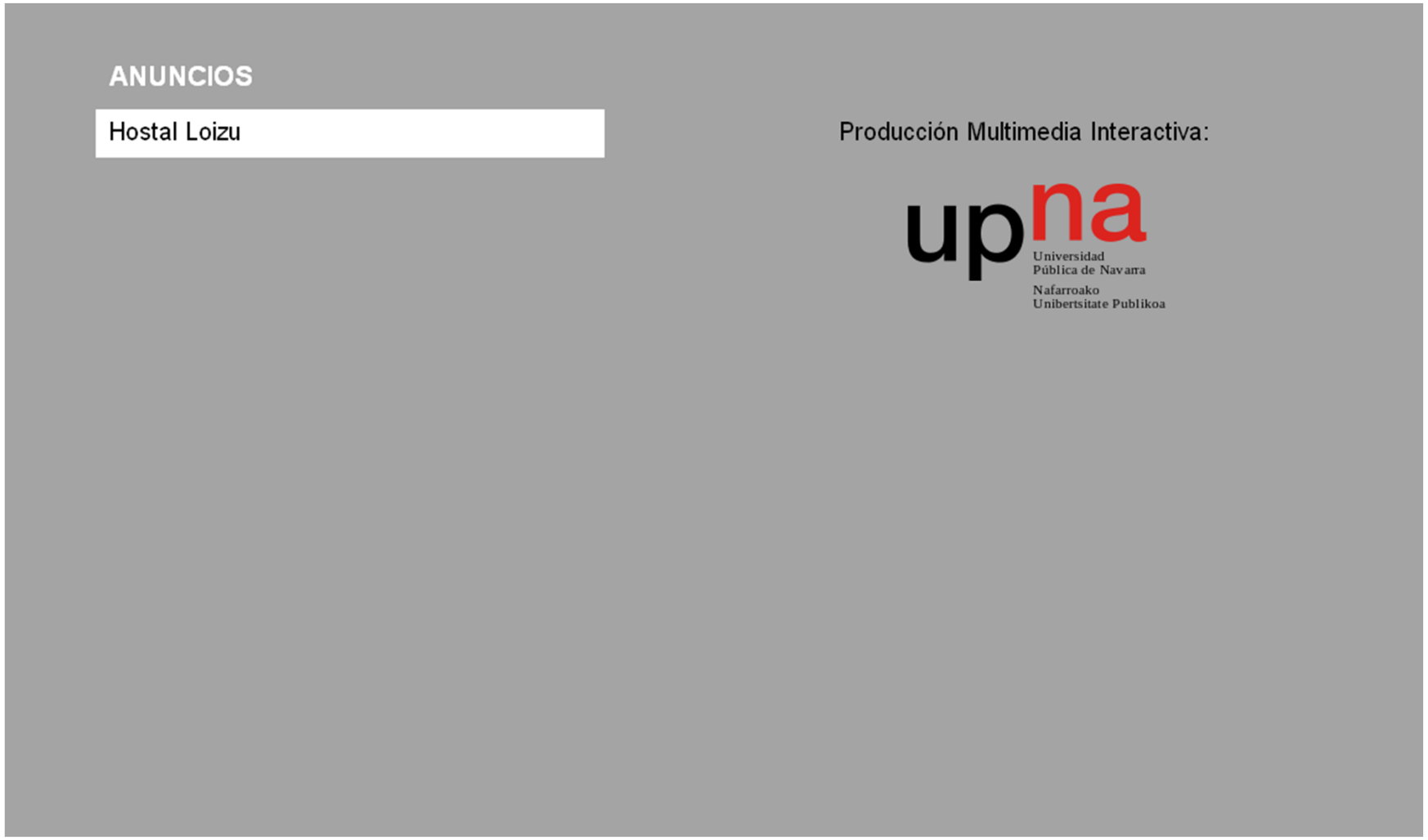
Práctica tema 6 Tecnologías e Instalaciones Vídeo

- **Se decide crear una práctica, que disponga de una aplicación HbbTV**
  - Menú principal con la serie de anuncios
  - Vídeo
- **Codificación vídeo**
  - Mp4 contenedor multimedia
    - Comúnmente vídeo
    - Subtítulos, imágenes..
  - Codificación h264
    - Alta compresión
    - Calidad de imagen
    - Sin incrementar complejidad de diseño

System Format	Video Format	Audio Format	Mime Type
TS	AVC_HD_25 AVC_SD_25 AVC_SP_25	HEAAC HEAAC2 HEAAC_MPS MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3 DTS	video/mpeg or video/mp2t
TTS	AVC_HD_25 AVC_SD_25 AVC_SP_25	HEAAC HEAAC2 HEAAC_MPS MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3 DTS	video/vnd.dlna.mpeg-tts
MP4	AVC_HD_25 AVC_SD_25 AVC_SP_25	HEAAC HEAAC2 HEAAC_MPS MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3 DTS	video/mp4
TS	MPEG2_SD_25 MPEG2_SP_25	MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3	video/mpeg or video/mp2t
TTS	MPEG2_SD_25 MPEG2_SP_25	MPEG1_L2 MPEG1_L2_MPS AC3 E-AC3	video/vnd.dlna.mpeg-tts

Table 1: A/V Media Formats for 25Hz video system

Práctica tema 6 Tecnologías e Instalaciones Vídeo



Práctica tema 6 Tecnologías e Instalaciones Vídeo

Hostal Loizu

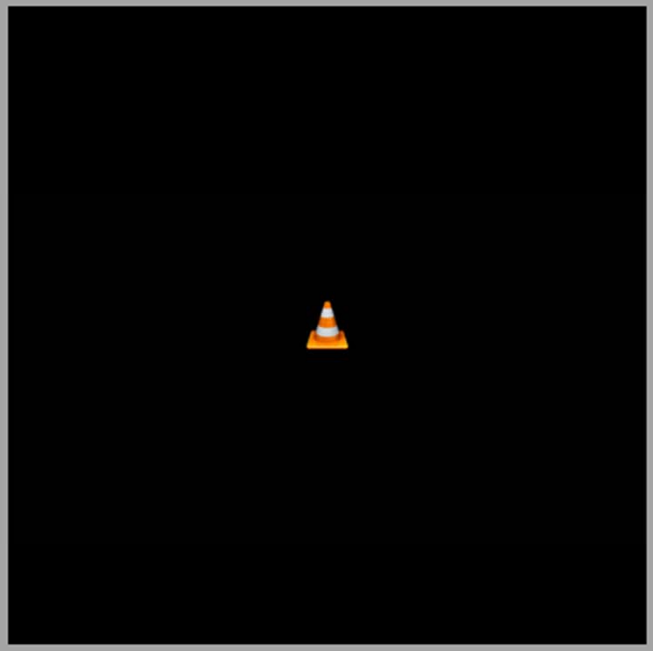
Enter para ver el vídeo

Enter para pause

Enter para continuar reproducción

Volver al menú de anuncios

El Hotel Loizu es un hotel rural de 3 estrellas y 27 habitaciones situado en la localidad navarra de Auritz-Burguete. Está a 3 km de Orreaga - Roncesvalles, muy cerca de este hito del Camino de Santiago, pero alejado del ruido y del trasiego continuado de peregrinos y visitantes.



Aplicación final: objetivos

- Una vez conocidas las tecnologías y realizadas las pruebas pertinentes era momento de crear la aplicación HbbTV final
- En primer lugar, había que decidir en que iba a consistir la aplicación :
  - Caso real
  - Posible aplicación dentro de un mercado

- Aplicación final va a ser la aplicación de un vídeo club que introduce su aplicación en el mercado de la televisión con el fin de ofrecer sus películas bajo demanda
- El objetivo de la práctica final era el de desarrollar dicha aplicación aplicando todos los conocimientos adquiridos durante el proyecto

Aplicación final: Tecnologías

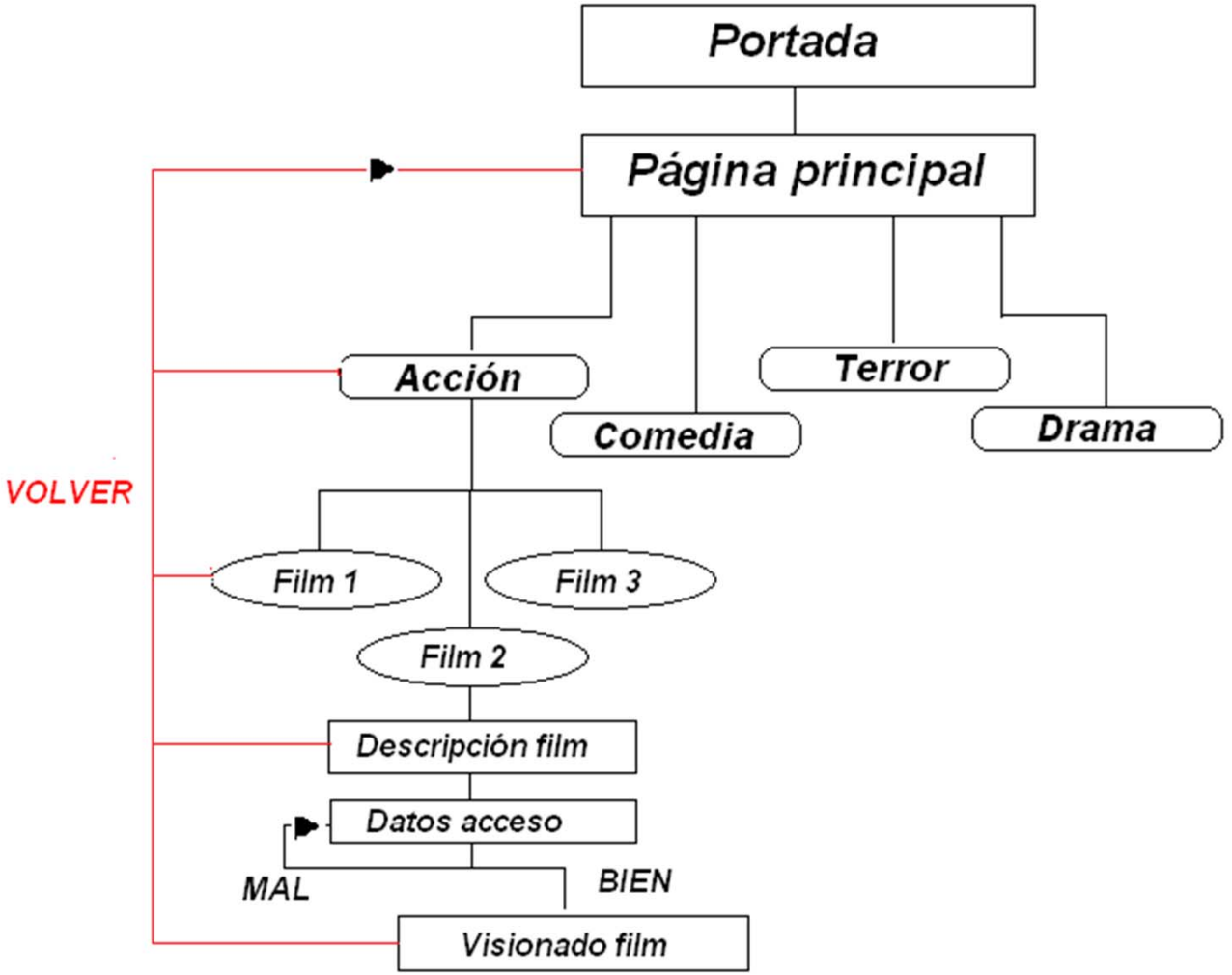
- **Para el desarrollo de esta aplicación final, se han utilizado todas las tecnologías previamente estudiadas:**
  - **Creación de nuestros archivos base, que serán la estructura de nuestra aplicación**
  - **Diferentes tipos de menús, tipos de letra, cuadros de imagen..(css)**
  - **Diferentes funciones dentro de la aplicación (javascript)**
  - **Profundidad de aplicación de más de un salto y relaciones entre diferentes páginas (php)**
  - **Creación de pantallas con buen aspecto agradables al usuario**
  - **Navegación adecuada a través del control remoto con funciones que aparecen en las aplicaciones reales del mercado**

Aplicación final: tecnologías

- Hasta el momento no se había hablado de otra de las tecnologías fundamentales a la hora de la creación de este tipo de aplicaciones: Bases de datos
  - Importancia de las operaciones con bases de datos en aplicaciones web, móviles, HbbTV
  - Tutorial de MySQL
    - Serie de vídeos con explicaciones y ejercicios resueltos
  - Tutorial de PhpMyAdmin (XAMPP)
    - Como crear bases de datos

	id	username	password
	1	usuario	usuario
	2	imanol	eslava
	3	pedro	pedro
	4	juan	juan

Aplicación final: Tecnologías



Aplicación final: Demostración

– A continuación se mostrará la aplicación creada

My Final APP

Conclusiones y líneas futuras

- Satisfacción personal
- Tecnologías (desconocimiento)
  - HbbTV
    - Html+Css
    - Javascript
    - Php
    - Bases de datos
- Tiempo dedicado a tecnologías
- Objetivo cumplido
- Problemas y mejoras
- Valoración personal de tecnología HbbTV

Gracias por su atención!

¿Preguntas?

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa